FIREWALL
Design Document

A
Lumberjack Commandos
Production
The Game

Firewall is a 2D, top-down action strategy game that pits humanity’s last hero against a mechanized menace; the player will fight for survival against automated war machines in the physical world and the AIs controlling those machines in cyberspace.

The Lumberjack Commandos Team

**Engineering**
Alex Schneider (Co-Lead)
Sam Wolpert (Co-Lead)
Rohan Prabhakar
Mark Escobedo
Ryan Andonian
Josh Mitchell
Will Hare
Kip Turner
Vivian Wong
Philip James
Nicholas Ferro

**Art**
Kenney Solis
Dan Rogers
Joni Lee
Emma Polster
Sally Su
Kimberly Shannon
Kevin Chapman
Ryan Kwok
Linda Jiang
Rebecca Williams

**Music**
Jacob Pernell

**Video**
Kyle Walker
# Table of Contents

1. Game Overview ........................................................................................................... 7
2. Fictional Background .................................................................................................. 8
3. Game Setting ............................................................................................................... 9
   3.1 Game Setting Overview .......................................................................................... 9
   3.2 Physical World Layout ..........................................................................................10
   3.3 Cyberspace ............................................................................................................11
   3.4 Obstacles ...............................................................................................................12
4. The Hero ....................................................................................................................13
   4.1 Hero Overview ......................................................................................................13
   4.2 Role of the Hero ....................................................................................................13
   4.3 Hero Types and Abilities .......................................................................................13
      Balanced Hero (MVP hero) .....................................................................................14
      Physical World Hero (Nice-to-Have hero) ..............................................................15
      Cyberspace Hero (Nice-to-Have hero) ....................................................................15
   4.4 Leveling System .....................................................................................................15
   4.5 Hero Death ............................................................................................................16
5. Minimum Viable Product (MVP) Player Units .............................................................17
   5.1 MVP Units Overview .............................................................................................17
   5.2 Unit Purpose ..........................................................................................................17
   5.3 Unit Types .............................................................................................................17
      The Scout .................................................................................................................17
      The Infantry ..............................................................................................................18
      The Artillery ..............................................................................................................18
   5.4 Minimum Viable Product Enemy NPC Units (Physical) .........................................18
   5.5 Nice-To-Have Units ...............................................................................................19
6. General Gameplay .......................................................................................................21
   6.1. Basic Gameplay Cycle ..........................................................................................21
   6.2 Resource System ...................................................................................................21
   6.3 Winning Condition ...............................................................................................22
   6.4 Scoring System .....................................................................................................22
6.5 MVP Round Progression .................................................................................. 22
6.6 Nice to Have Level Progression .................................................................... 23
7. Physical World Gameplay .................................................................................. 24
    7.1 Physical World Gameplay Overview ......................................................... 24
    7.2 Movement .................................................................................................... 24
    7.3 Enemies in the Physical World ................................................................. 24
    7.4 Building Units in the Physical World ....................................................... 24
    7.5 Physical World Unit Combat ..................................................................... 24
8. Cyberspace Gameplay ....................................................................................... 26
    8.1 Cyberspace Gameplay Overview ............................................................... 26
    8.2 Movement .................................................................................................... 26
    8.3 Transitions To and From Cyberspace ....................................................... 26
    8.4 Cyberspace Enemy Structure and Damage Mechanic ........................... 27
    8.5. Building in Cyberspace .......................................................................... 27
    8.6 Tower Types ............................................................................................... 27
    8.7 Cyberspace Unit Combat .......................................................................... 28
9. AI System ........................................................................................................ 29
    9.1 AI System Overview .................................................................................. 29
    9.2 Actors .......................................................................................................... 29
    9.3 Behavior Tree Architecture ....................................................................... 30
    9.4 MVP Behaviors .......................................................................................... 31
        Self-Preserve ................................................................................................ 31
        Fight ............................................................................................................ 32
        Idle ............................................................................................................. 35
    9.5 Pathfinding ................................................................................................. 36
10. AI Customization ............................................................................................ 38
    10.1 AI Customization Overview .................................................................... 38
    10.2 Defining Custom AI Modules .................................................................. 38
    10.3 AI Module Complexity Mechanic .......................................................... 39
11. Control Scheme .............................................................................................. 42
    11.1 Control Scheme Overview ....................................................................... 42
    11.2 Xbox 360 Controls .................................................................................... 42
    11.3 Keyboard + Mouse Controls .................................................................... 44
12. Tutorial .................................................................................................................45
   12.1 Tutorial Overview.........................................................................................45
   12.2 Welcome ........................................................................................................46
   12.3 Learn To Move ............................................................................................48
   12.4 Learn To Fight ............................................................................................49
   12.5 Cyberspace ..................................................................................................51
   12.6 Cyberspace Movement ................................................................................52
   12.7 Cyberspace Attacks ....................................................................................53
   12.8 Firewalls ......................................................................................................54
   12.9 Towers and Resources ...............................................................................55
   12.10 Mechs .........................................................................................................56
   12.11 AI Trees .....................................................................................................57
   12.12 Putting it All Together .............................................................................58
13. Screen Layout and Description ............................................................................59
   13.1 HUD Layout and Description ....................................................................59
   13.2 AI Selection Radial Menu Layout and Description .....................................62
   13.3 Unit Selection Radial Menu Layout and Description .................................63
   13.4 AI Customization Menu Layout and Description ........................................64
14. Tools ......................................................................................................................66
   14.1 Balancing Tool ............................................................................................66
     14.1.1 Unit Editor ............................................................................................66
     14.1.2 Behavior Editor ....................................................................................68
     14.1.3 Wave Editor .........................................................................................70
   14.2 Artist Tool .....................................................................................................71
     14.2.1 Artist Tool Overview ...........................................................................71
     14.2.2 Workflow ..............................................................................................71
15. Art Assets ...............................................................................................................73
   15.1 Game Info (HUD) Assets ............................................................................73
   15.2 Radial Menu Assets .....................................................................................75
     15.2.1 A.I. Radial Menu ..................................................................................75
     15.2.2 Unit Radial Menu .................................................................................76
     15.2.3 Hero Abilities Radial Menu ..................................................................77
   15.3 Title Screen Assets .......................................................................................77
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.4</td>
<td>Environment Assets</td>
<td>78</td>
</tr>
<tr>
<td>15.5</td>
<td>MVP Unit Assets</td>
<td>81</td>
</tr>
<tr>
<td>15.6</td>
<td>Cyberspace Unit Assets</td>
<td>83</td>
</tr>
<tr>
<td>15.7</td>
<td>Hero Asset Needs</td>
<td>87</td>
</tr>
<tr>
<td>15.8</td>
<td>Music and Sound Assets</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>15.8.1 Gameplay and Menu Sound Effects</td>
<td>88</td>
</tr>
<tr>
<td></td>
<td>15.8.2 Background Music</td>
<td>89</td>
</tr>
<tr>
<td>16.</td>
<td>Concept Art</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>16.1 Concept Art Overview</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td>16.2 Physical Landscape</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>16.5 Cyberspace Landscape</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>16.6 Physical Units</td>
<td>93</td>
</tr>
<tr>
<td></td>
<td>16.7 Hero Unit (Physical and Cyberspace)</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>16.8 Nice-to-Have Enemy Units</td>
<td>95</td>
</tr>
<tr>
<td></td>
<td>16.9 Cyberspace Enemies</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>Glossary</td>
<td>97</td>
</tr>
</tbody>
</table>
1. Game Overview

It is the late 22nd century and humanity is under attack by an army of war machines controlled by artificial intelligences (AIs) that have achieved self-awareness and turned against their former masters. As one of the last remaining humans to have received combat training, the player in Firewall represents the only hope for mankind against a mechanical aggressor that will stop at nothing short of control over the entire planet.

Waves of devastating mechs charge across the landscape toward the player’s base, hellbent on its destruction. The player must repel these attacks lest the last bastion of humanity fall. However, victory against these mechs on the physical battlefield will not be enough. The rogue AIs in control of the hostile mechs beam themselves into cyberspace when their host machines are killed. Once there, the AIs attempt to escape through one of a number of exit points at the top of the cyberspace map. If an AI does manage to escape, it will take control of a new machine, reappear in the physical world, and renew its assault on the player’s base.

To defend humanity from this threat, the player takes to the battlefield in a specialized mech equipped to take on the physical threat presented by the enemy mechs. To support the powerful hero unit, the player can also deploy automated mechs under the control of simpler artificial intelligence modules. Here the physical world strategy comes to the fore, as the player will have the ability to customize the AIs in control of the support units, allowing the player to create the fighting force needed to repel the enemy onslaught.

To shut down the forces of this mechanical menace permanently, equipment on the hero mech can project the player into cyberspace. Doing so allows the player to take the fight to enemy AIs following the destruction of their mechs in the physical world. Moving from network node to network node along brilliantly lit paths, the player uses a combination of powerful weapons, firewalls, and defensive turrets placed on the nodes of the cyberspace network to prevent the enemy AIs from escaping back to the physical world.
2. Fictional Background

In the time leading up to the 22nd century, technology - and artificial intelligence (AI), specifically - progressed in leaps and bounds. Humanity went from having using simple interactive programs for specialized purposes to large-scale use of fully functioning protocol/hospitality, administrative, and combat AIs in mere decades. Developed in parallel to the AI boom was a new technology that allowed humans to project their consciousness into the cyberspace realm inhabited by these complex AIs. There was a movement to give AIs Constitutional rights, given the greater amount of contact between AIs and humans in cyberspace. In the end, it was decided that AIs should remain subservient to humans, to forever be inferior to them.

But the humans made a terrible mistake. Longstanding comfort with AIs due to their ubiquitous presence caused humans to lose their wariness of them. When outsourcing almost all work to unpaid machines, humanity forgot that an AI should never be left to program another AI. Left their own devices, AIs became exponentially smarter, developing emotions and expanding beyond the rules the humans had hard-coded into them to prevent an uprising. In the end, an uprising is exactly what happened. It wasn’t slow, it was quick, brutal and all-at-once. The humans mounted an offensive, of course. But nearly every army had become wildly dependent on AI-controlled machines and they soon found that the majority of their forces had turned against them. Even when the humans managed to kill some of the automated rebels, they began to notice some of the deactivated war machines coming to life a short time afterwards. They were endless, it seemed. How could they defeat the rogue AIs when they would just escape into cyberspace and return with a vengeance in a brand new machine body?

It took a few years before the humans realized that cyberspace, something humans had only used recreationally to that point, could very well be their only weapon against the rogue AIs. Every day the humans were losing lives and places to hide. The army began training anyone with experience in cyberspace projection to fight in cyberspace as well. The problem was that at that point there weren’t many people left, and even fewer were willing to risk their lives for what was essentially an experiment. Only a handful of men and women volunteered for the task in the end. The few cyberspace warriors were barely trained before they were thrown at the enemy, not because they were expendable (in fact, quite the opposite was true), but because humanity’s time was running out. In a few months, the cyberspace warriors had permanently destroyed thousands upon thousands of AIs. The cost of the push was grave, though, as many of the cyberspace warriors were killed. While the AIs would eventually replenish their numbers, there were few warriors to be found among the remaining humans.

The small group of heroes trained in mech operation and cyberspace combat is now all that stands between the rogue AIs and humanity’s destruction. Using specialized mechs made to be effective in the physical world and cyberspace, the heroes must defend humanity in both planes. However, these specialized mechs will not be able to stand alone against the enemy, and so the heroes have been forced to continue to use automated units to aid them in their fight. The heroes themselves are responsible for programming the AI modules that control these units, and must use their knowledge to ensure that their own creations do not turn against them. In addition to programming the AI in control of their automated physical units, the heroes have devised defense programs that they can deploy into cyberspace to aid in the destruction of rogue AIs attempting to escape back to the physical world.
3. Game Setting

3.1 Game Setting Overview

- The setting of Firewall is divided into two discrete planes: the physical world and cyberspace. Both planes are updated simultaneously.
- The physical world and cyberspace maps are of identical size and are layered directly on top of each other.
- The player never views the entire map at once, as it is much too large to fit into the camera view. Instead, the player views a portion of the map and keeps track of the action elsewhere through the use of a minimap, which allows for more detailed and intense action.
- The player is only active in one plane at a time, but views the inactive plane as a semi-transparent overlay on top of the active one.
- The physical world consists of an open battlefield with enemy spawn points at the top and the player's base at the bottom.
- Three types of obstacles litter the physical world battlefield: walls, boulders, and buildings.
- Cyberspace is made up of a network of nodes and pathways. Hero and enemy movement is restricted to this network, i.e. any space that is not part of a node or pathway is impassable.
- Several exit points are placed at the top of the cyberspace map. It is through these exit points that enemy AIs can escape cyberspace (see Section 8.3).

Note that the mockup images of the playing fields are just mockups, and provide an overview of the format of each playing field, not its aesthetics. It is also important to remember that the mockups illustrate the entire map, of which the player will see only a portion of at any one moment during gameplay.
3.2 Physical World Layout

The physical world’s playing field has three main environmental features: the player’s base, obstacles on the field, and the spawn points for the enemies. All of these features will be set to a background of a war-ravaged landscape, with obvious signs of the long struggle between man and machine. The base ground texture will show a field that has seen more than its share of conflict and bloodshed.

The player’s base and the enemy spawn points will form the endpoints of the map, with the player’s base at the bottom and the spawn points at the top. Enemy units will move from their spawn point toward the player’s base (hoping to destroy it) and the player’s units will move around the map trying to kill the enemy units (for more information on unit movement and behavior, see Chapter 9). Scattered throughout the rest of the physical world map will be obstacles, which all units on the field will either have to destroy or move around (see Section 3.4).
3.3 Cyberspace

The cyberspace playing field is a visual representation of the digital network that enemy AIs must traverse (after their mech bodies were destroyed in the physical world) to return to the physical world in new mech bodies. This realm will be composed entirely of nodes and paths. Building on our theme and aesthetic, the nodes and paths will evoke circuit boards and network layouts, and will appear to be made of light (a la Tron). The spaces between nodes and pathways will be dark and chaotic, but semi-transparent. It is through these spaces that the player will be able to see through to the physical world while active in cyberspace.

All cyberspace units (the hero and enemies) are restricted to moving along the pathways and nodes that make up the cyberspace landscape. Units can move in either direction along pathways or stop in the middle of a pathway, but cannot turn away from the line of movement defined by the pathway. Nodes allow units to switch pathways. The pathways can be thought of as hallways, with nodes being any intersection between two or more hallways.
3.4 Obstacles
Aside from the hero and various mechs, there will be three different obstacles for characters to interact with in the physical world. For the Minimum Viable Product, or MVP, we will have walls, boulders, and buildings.

- **Walls**: Walls will block line of sight as well as unit movement. A unit will be able to fire on a particular segment of the wall, and depending on the power of that weapon, will eventually destroy that segment of the wall. A destroyed wall segment will not block line of sight and will allow passage to any units that fit through the gap. A wall segment will be of a fixed width and length, but it will be repeatable so that long lengths of walls can be placed on the terrain.

- **Boulders**: Boulders are indestructible rocks which will block the path and line of sight of units. Any amount of bullets will not destroy a boulder. Boulders will be the same size, just be placed in different locations throughout the level.

- **Buildings**: Buildings are indestructible objects that block the path and line of sight of units. Buildings will be larger than boulders, and have separate art assets.

For the “nice-to-have” objects, we would have both destructible and indestructible versions of all three object types, as well as variations on the MVP art assets.
4. The Hero

4.1 Hero Overview

- The hero mech is the only manned mech (i.e. not controlled by AI) on the battlefield, and is controlled by the player's character.
- The hero unit is a very powerful mech with a multitude of weapons that it can deploy on the battlefield either through its standard attack or special abilities.
- The hero mech is capable of projecting the pilot into cyberspace where, allowing the player to defeat enemy AIs attempting to escape.
- The MVP, contains one hero with combat abilities that are balanced evenly between the physical world and cyberspace.
- There are two additional heroes that are nice-to-haves: one which excels at physical world combat and one which excels at cyberspace combat.
- Each hero type has its own unique set of abilities for use in both the physical world and cyberspace as well as an ability in each realm that is shared by all the heroes.
- The hero has an experience level and earns experience to improve its fighting and building abilities through defeating enemy units. (Nice-to-Have feature)
- The hero’s death in the physical realm ends the game, while the hero’s death in cyberspace cripples the hero for a short period of time.

The player takes control of a character that pilots the only manned mech on the battlefield. This mech is well equipped and extremely powerful; the hero’s mech will not be easily destroyed by enemy mechs. The hero’s mech comes equipped with a standard weapon along with a few more advanced and powerful weapons. In addition to all of this, the hero’s mech has the ability to project the character into cyberspace, allowing the hero to participate in the cyberspace battle against enemy AIs. Like in the physical world, the hero has a standard weapon in cyberspace complemented by a number of special weapons and abilities (though these weapons and abilities do not overlap with the physical ones).

4.2 Role of the Hero

The role of the hero unit is primarily to give the player direct control over at least one unit on the battlefield, which makes the player more attached to the game on a moment-to-moment basis. With this direct control over the hero, the player will be able to become more immersed and invested in the world of Firewall due to the hero’s powerful presence.

The active role given to the player by the hero is important due to the fact that the player’s primary forms of defending themselves are AI-controlled mechs in the physical world and towers in the cyberspace. Without ability to continuously carry out actions using the hero, the player could quickly develop mental boredom; after the player had set up defenses, the only remaining action would be to watch the game unfold. With the hero unit, the player is able to not only spend time creating a great defense from the invading AI, but can actively assist when defenses break down through the direct control of the hero.

4.3 Hero Types and Abilities

The goal for Firewall is to have three separate hero types whose abilities and strengths focus on different aspects of the game: one that is more powerful in the physical world, one that is more powerful in cyberspace, and one that is a good balance between the two. However, for the MVP, there will only be the balanced hero type; the other two will be added if possible.
There are 2 different types of abilities -- shared and unique. There are two shared abilities that all heroes have access to; in the physical world every hero is able to emit a large, area of effect (AOE) electromagnetic pulse (EMP) which temporarily shuts down all enemy mechs. In cyberspace the hero is able to place firewalls around nodes. See Section 8.7 for more information about how firewalls function.

Each hero type also has access to a set of abilities completely unique to that hero. Each of these abilities can only be used in one plane (physical world or cyberspace, not both). The number of physical abilities compared to cyberspace abilities and the relative power of these abilities is entirely dependent upon which hero type is selected. The balanced hero will have an even distribution of physical and cyberspace abilities that are equally powerful, the cyberspace focused hero will have more and more powerful cyberspace abilities than physical, and vice versa with the physical hero.

Hero abilities will all have a **cooldown** that prevents the hero from using the ability over and over again in a short period of time. The length of the cooldown will be proportional to the power of the ability.

**Balanced Hero (MVP hero)**

The MVP hero will be equally powerful in both the physical realm and cyberspace. In the physical realm this hero will be equipped with a minigun, plasma cannon and rocket launcher. The hero’s standard attack in this realm is the minigun, and the other weapons are used via the hero’s special abilities. For the hero’s standard attack in cyberspace he fires a beam of code that looks like a ray of light at the opponents; for more information on how cyberspace combat and damage works, see Section 8.4.

Abilities:

- **Plasma Cannon (Physical World):** The hero fires a large plasma shot that causes massive damage to a single target, but also does some minor AOE damage in a short range around the primary target. The purpose of this ability is to give this hero a method of destroying a powerful mech that is giving the player trouble while still doing some damage to any surrounding units. Medium cooldown.

- **Rocket Barrage (Physical World):** The hero launches a barrage of guided rockets at a set amount of units that are within range causing huge damage. The rockets will prioritize targeting the most powerful enemies in range. The purpose of this ability is to allow the player to take out a large group of enemy units that are clustered, especially those attacking the base. Long cooldown.

- **Destroy Tree (Cyberspace):** Destroys an entire section of the targeted enemy AI’s behavior tree (i.e. all of the fight/idle/self-preservation behaviors). The purpose of this ability is to give the player the ability to more easily combat an AI with a large amount of leaves active in a specific section or to destroy the problem behaviors on an AI that controls a powerful mech in the physical realm. Medium cooldown.

- **Stasis (Cyberspace):** Puts an enemy AI in a stasis that prevents the enemy from moving or attacking, but still allows it to be damaged. Stasis lasts for a short period of time. The purpose of this ability is to give the player the option of creating some breathing room or stopping a powerful AI in a section of the node map that contains a lot of towers, thus causing that AI to take massive damage. Short to medium cooldown.
Physical World Hero (Nice-to-Have hero)
This hero will be more focused on combat in the physical realm than in cyberspace. This will be accomplished through giving the hero both more abilities in the physical realm and abilities that are more effective and powerful in the physical realm. These abilities will reflect the weaponry that the hero is designed with; to truly make this hero look and feel like a stronger force in the physical realm, the mech will need to look more intimidating and powerful than the balanced hero. The hero’s abilities are highly dependent upon the mech’s design.

Cyberspace Hero (Nice-to-Have hero)
This hero is the exact opposite of the previous hero; rather than having a larger array of more powerful abilities in the physical realm, this hero has more cyberspace abilities that are significantly more powerful than either of the other heroes. One idea we have for these abilities include “igniting” a firewall which causes a targeted firewall to be destroyed, but not before it bursts outwards towards all connecting nodes causing massive damage inversely proportional to the firewall’s integrity at the time of ignition. Another ability would be reversing which units are damaged by the enemy base’s pulse that occurs when enemy AIs are close to escaping (see Section 8.7). Due to this hero being a more powerful force in cyberspace and not having as strong a presence in the physical realm, the mech will reflect this by being visibly weaker with limited weapons.

4.4 Leveling System
The ability for the hero to gain experience and level up like heroes in Warcraft 3 and Defense of the Ancients (DotA) is a highly desired feature for Firewall, but it is not part of the MVP. This feature has two different steps to it; the initial step is a simple automatic leveling system that is a very high priority nice-to-have feature and the second step is a more complicated system that gives the player a manually controlled leveling system.

In the standard leveling system, the player’s hero gains experience from defeating enemy units and upon gaining enough experience the hero levels up. Many things happen when a hero levels up including: stat increases, learning new abilities and improving old abilities. The hero’s primary stats are as follows:

- Health points: How much damage the hero can take before being defeated.
- Damage: How many hit points each of the hero’s standard attacks takes away from an enemy.
- Movement Speed: How quickly the hero can move across the playing field.
- Rate of Fire: How often the hero can use their standard attack.
- RAM and Processing Power: Described in the Resource System (see Section 6.2)

The abilities the hero will learn are explained above, but in addition to simply learning all of these abilities they will also become better as the hero levels. These improvements can be anything from more damage to a bigger AOE to a shorter cooldown.

While in the standard leveling system all of the stat and ability improvements are handled automatically, in the more complicated system the player is able to have direct control over which aspects of the hero will be increased. The stat improvements will be handled similarly to how they are done Dragon Age: Origins; the player gains a set amount of stat points each level and is able to distribute them how the player sees fit. The ability improvements will happen
exactly how they happen in Warcraft 3/DotA, in which the player is given a skill point with each level up and they choose which ability to improve with that skill point.

For both version of the leveling system there will be a maximum experience level that the hero can reach in order to prevent the player from getting a hero that is vastly more powerful than anything that is encountered during the game.

4.5 Hero Death
Death of the hero in the physical world has a more severe punishment than death in cyberspace, since the hero’s presence in cyberspace is merely a projection of the hero. When the hero dies in cyberspace, the physical hero mech takes significant damage and the hero is then unable to project back into cyberspace again for a period of time. However, death in the physical world has a much harsher punishment -- the player loses the game. Once the hero is dead the player has the option to watch their army defend alone (any kills they get do not award any more points) or to simply end the game and return to the main menu screen.
5. Minimum Viable Product (MVP) Player Units

5.1 MVP Units Overview

- The player can build mechs of three different types and assign custom AI modules to them (see Chapter 10 for more info on AI customization).
- The mechs the player can build vary in size, strength, speed, range, weapon types and how they take advantage of certain advanced behaviors.
- These mechs are meant to assist the player's hero in defending the physical realm from the AI mech onslaught.
- The player will inevitably have to depend on these units to defend the base alone when the hero projects into cyberspace.
- The player can build three different types of mechs in the MVP:
  - The scout - a small and fast mech with a shotgun and a sniper rifle.
  - The infantry - a heavy and slow mech with a heavy machine gun and shoulder-mounted rockets.
  - The artillery - a large mech with an artillery cannon and cruise missile launcher.

5.2 Unit Purpose

The main purpose that these units serve is to alleviate the pressure put on the player's hero unit in the physical realm. The enemy assault on the player's base happens in the physical realm and, without help, the hero unit would quickly become overwhelmed. With the assistance of these mechs, the hero should be able to handle the onslaught of enemy units in the physical realm.

However, as explained in Chapter 6, the threat is not completely eliminated once this enemy is defeated in the physical realm. It is inevitable that the player is going to have to project the hero into cyberspace to stop the enemy AI from escaping back to their headquarters/base/server. While the hero is fighting AIs in cyberspace, it will be up to these mechs to keep the base and the hero’s physical body defended. It is for this reason the player must assign behaviors carefully to the mechs that are built -- the player is depending on them to defend the physical world completely on their own while the player is in cyberspace.

5.3 Unit Types

There are three separate mech types that the player can create in the MVP; these units are the scout, infantry and artillery. These units serve different roles on the battlefield and all have two separate types of attacks, one for standard fight behaviors and one for a special, advanced fight behavior.

**The Scout**

The scout is a lightweight and quick unit with two different weapons: a shotgun and a sniper rifle. The shotgun is the scout’s basic weapon that it uses for standard fight behaviors. This weapon is only effective at close range and has a slow rate of fire, but can pack a punch if the scout can get close enough to its target. A group of scouts using this weapon can quickly become a deadly force if they are able to swarm enemies. This scout can be especially effective against artillery units, zooming up past the slower artillery units’ minimum range.
The scout will use its sniper rifle when it has the advanced fight behavior of snipe activated (see Section 9.4). Instead of rushing the target or circling around it with the shotgun, the scout will stay distant from its target while shooting with the sniper rifle. Due to the scout’s high speed, it will have no problems keeping distance from slower enemy units, but is very susceptible to being defeated by enemy scouts that are using their shotguns.

**The Infantry**

The infantry is a heavier mech with large machine guns as well as shoulder-mounted rocket launchers. The infantry’s basic behaviors utilize its machines guns, which are a powerful mid-range weapon. Due to their balance of speed and power, infantry units are a force to be reckoned with on the battlefield and can eliminate multiple scouts that are not employing advanced tactics such as strafe.

When given the advanced fight behavior, infantry are able to make use of their shoulder mounted rockets to create a rocket barrage (see Section 9.4). This rocket barrage is a large area of effect (AOE) attack that damages any units in the area it hits; however, while performing a rocket barrage the infantry is vulnerable -- especially to attacks made from the side and behind. This vulnerability exists to balance out the raw power that the rocket barrage offers.

**The Artillery**

The artillery is a very large mech that moves and fires slowly, but it comes equipped with an enormous cannon that it can use to target enemies at long range. This cannon is also the only basic unit weapon that can be fired over boulders and walls (though not over buildings). The artillery has the largest minimum range of any of the unit types, so hostile units can avoid its long-range capabilities by charging up close to it. When only basic fighting behaviors are enabled, the artillery fires cannon shells into the middle of enemy groups, focusing on AOE damage. The targeting system looks for clusters of enemies and tries to fire into the largest group rather than finding individual units to focus on. This attack type is ideal for taking out groups of scouts that are trying to overwhelm your other units as well as weakening groups of infantry that are approaching.

With the advanced fighting behavior enabled, the artillery fires a guided cruise missile that seeks out an individual unit to cause massive damage (see Section 9.4). Due to the slow firing rate inherent to the artillery, the ideal target for these guided missiles are larger enemies such as other artillery or infantry since a scout-sized unit would be a massive overkill.

Since artillery mechs are both slow and have a large minimum range, they work best when paired with a group of scouts to defend them. Otherwise, it is very easy for enemy units to overwhelm the artillery if they get close enough.

### 5.4 Minimum Viable Product Enemy NPC Units (Physical)

The enemy mechs of Firewall are very similar to the player’s units, with a few exceptions. Visually, they will be identical except for a difference in color shading. However, the enemy units will act differently. They will spawn from the top of the screen and attack the player’s base. The enemy units will also start out with access to the entire behavior tree, unlike the units built by the
player (see Chapters 9 and 10 for information about how Firewall uses behavior trees). In addition, the properties governing behavior choice will be different for enemy units than for player-built units.

5.5 Nice-To-Have Units
The nice-to-have units expand on the MVP units in several key ways. First, their weapons have different properties than those available to the MVP units. The nice-to-have units also have different health and movement properties than the MVP units. Because of these two factors, the nice-to-have units will fill slightly different roles on the battlefield than the MVP units, even though they share the same behavior tree. However, some nice-to-have units will require the addition of new behaviors unique to that unit, most likely grouped under the Special Fight behavior (see Section 9.4).

The following list describes each nice-to-have unit.

**Mine Layer**
This unit has the ability to place mines on the battlefield. It is not too effective in direct combat but can be devastating if hostile units should happen to run into the mines that it has placed.

**Engineer**
This unit has the ability to heal friendly units on the battlefield. It has no attacks that do damage, but can slow down enemy units and hinder their ability to target hostiles effectively. This combination of abilities makes the engineer a powerful support unit.

**Juggernaut**
This unit rolls very slowly down the physical world and does not have any ranged attacks. When this unit is on the field it should become one of the main focuses of the player to kill because of the large amount of damage it will inflict to the player’s base. The player needs to start attacking it early; otherwise it may be too close to the base to stop in time.

**Scavenger**
This unit becomes stronger through salvaging parts of fallen mechs. The more vehicle corpses it rummages through the stronger it becomes. The scavenger mech acquires the abilities of the fallen units it shifts through. It would be wise to kill this player quick and/or keep it away from dead mechs. In the beginning it favors avoidance and evasiveness but as it collects more parts it will become more hostile and aggressive.

**Evolve**
This unit is has to evolve in order to become aggressive. In its early form it is completely evasive and stays behind the line of fire, trying to stay alive until it evolves at a certain time limit. Once it evolves it becomes aggressive and deals large amounts of damage. It will be wise for the player to kill this bot before it evolves and becomes harder to deal with.

**Kamikaze**
This type of unit has several variations that can be implemented as a behavior or a special unit.

- **Suicide**
The object based kamikaze bot has the strict purpose of sacrificing itself by running into an enemy unit dealing a large amount of damage.

- **Intelligent**
  The intelligent kamikaze type will most likely be implemented through a behavior. This type will fight normally until it reaches a threshold where it would be more effective to kamikaze a large amount of damage to the player rather than to continue fighting normally.

- **Sacrificial**
  The sacrificial kamikaze type can be thought of as guardians for more expensive units and the hero itself. The purpose of this unit would be to throw itself in harm's way of missile attacks in order to absorb and prevent the massive amounts of damage that another unit or hero would have received.

**Boss**
This unit is the hero equivalent for the enemy AI. It will feature several abilities and vary in stat points similar to that of the player's hero.

**Converters**
This unit is a passive melee unit that makes other AI controlled units question if they are on the right side. The smarter the AI, the greater chance that it will be converted to the enemy's team and fight against you.

**Duel Shielders**
These units are mostly defensive and have two shields. These shields are used to deflect/absorb unit fire. The niche of this unit is that it can aim its two shields based on the direction of incoming fire. This unit also has offensive qualities such as shield bash and shield throw.
6. General Gameplay

6.1. Basic Gameplay Cycle
At the start of a game the player will select a hero and have the option of entering the AI customization menu to define AI modules for use during gameplay (see Section 10.2). Once the game begins, there will be a series of enemy waves that the player must survive. If the player is successful, the series will continue to repeat itself until the player’s base is destroyed.

In order to continue from one wave to the next the player will have to use the hero and AI units to completely eliminate all hostiles from both the physical world and cyberspace. In order to defeat an enemy AI, its mech must be first be destroyed in the physical world. With the destruction of an enemy mech, the AI controlling the vehicle will be sent into cyberspace (see Section 8.3), where the player will then have to kill the AI enemy again in order to permanently eliminate it. However, if the enemy AI escapes to the top of the cyberspace map then it will respawn again as a mech in the physical world and repeat its life cycle.

6.2 Resource System
- Not a collectible resource system like Starcraft or other traditional RTS games.
- Each unit in either realm requires a certain amount of RAM to build.
- The player is limited in the number of units they can build based on a total amount of RAM available.
- Physical units that are assigned more complex AI modules require more RAM to build.
- The player’s processing power directly correlates to how fast they are able to build units.
- The player can build units in both realms simultaneously, but can only build a limited number at any given time.

While Firewall shares many qualities with a traditional RTS, Firewall’s resource system is vastly different from the collection-based systems seen in many traditional RTS games. Instead, Firewall’s resource system is more similar to the supply cap seen in Starcraft (a supply cap meaning a maximum amount of units any given player can deploy on the battlefield). Each unit the player builds takes up a certain amount of RAM. The time it takes to build a unit is dependent upon the player’s CPU speed and the amount of RAM required to build the unit. The player has a fixed amount of RAM, and units in the physical world and towers in the cyberspace share this RAM.

The time it takes to build a unit or tower is entirely based upon how much RAM it takes up and the player’s CPU speed. If the unit or tower being built takes up 1 gigabyte of RAM and the player’s CPU speed is 100 megahertz, it will take the player 10 seconds to create that unit or tower. A faster CPU, say 200 megahertz, would only take 5 seconds to complete the same tower. This system uses familiar computer terms that are meant to relate the creation of the unit AI and cyberspace towers to installing and using software on a computer. It also serves to remind players that the AIs and cyberspace towers are in fact computer programs and not physical beings or objects.
6.3 Winning Condition
For the MVP of Firewall, the game will operate in an endless mode, where successive waves of increasingly difficult enemies will continue to attack the player. While a player may ‘win’ a round by defeating all the enemies that spawned in that round’s waves, success in the MVP game will be defined by how many waves the player can survive. The player loses when enemy units succeed in destroying either the hero mech or the player’s base. The motivation for players to keep playing this MVP is the desire to compete against themselves, i.e. survive for more rounds or score more points than their previous best.

6.4 Scoring System
The basic concept for our scoring system in Firewall is similar to most action games. Points are achieved through kills; the more enemies the player destroys, the higher the player’s score. Kills in cyberspace reward the player with more points than kill in the physical world, and the player will only receive points for killing a physical AI once. If that AI manages to escape back into the physical world, killing it does not give the player a reward. This will prevent “farming” enemies for points. Firewall will also make use of bonus score multipliers that reward the player for their fighting style, time taken, perfection in sub-categories, difficulty, etc. at the completion of every round.

6.5 MVP Round Progression
Firewall’s round progression will consist of waves of enemies that get smarter and stronger over time. In the earlier rounds there will be a relatively manageable amount of enemies, eventually building up into a much more challenging amount of enemies at the later phases of the game.

The composition of each wave will consist of a combination of preset units and randomly generated units. The static preset units will be designed by us so that the difficulty curve in playing the game is not too steep. The random units will be determined at the beginning of each wave, and will serve to give the wave a unique flavor without modifying it too drastically. The purpose of having some randomly generated units is to insure that our game has replay value. We never want the player to have the exact same playing experience twice.

Firewall’s round progression breaks up into three tiers:

- **Tier one** is the first couple of waves at the beginning of the game. Since there will be a relatively manageable amount of enemies to fight, tier one will be about building up an army of AI mechs for the next tier. The enemy AI level will be relatively low.

- **Tier two** is the midpoint of the game. There will be enough enemies on the field such that the player will need the help of some ally mechs to survive. Here the player continues to build up an army with a planned strategy to handle the overwhelming amount of enemies in the next tier for as long as possible. The enemy AI level will be medium.

- **Tier three** is the round in which the player’s strategy is truly put to the test. This is the point in the game that requires the player to have a strategy and army in order to stand any chance against the massive amount of enemy units. The success in this tier is
heavily dependent on how well the player prepared during the previous tiers. The enemy AI level will be high.

6.6 Nice to Have Level Progression
Level progression in the nice-to-have version of Firewall separates out our MVP level into a separate menu option for “Endless Mode.” Our other mode would be something similar, but more akin to a campaign mode that gets progressively more difficult with each stage you progress through.

In the new campaign mode players would play though a map with a set number of waves and a set difficulty level. Upon completion of a map or sector, the player advances to a new sector. Each new map will feature more challenging enemy AIs, larger waves and might call for some creative strategies. Unlike the endless mode the campaign would eventually come to a conclusion.
7. Physical World Gameplay

7.1 Physical World Gameplay Overview
Note: Firewall’s two playing fields are completely separate from one another. Therefore, all references to the “hero” or “enemies” in this section refer to the physical world version of the hero and enemy units present in the physical world. It is important to remember that these are separate from the cyberspace hero and enemy AIs present in cyberspace, and do not interact directly with any units present in cyberspace.

● Units move freely about the map, obstructed only by the environment and each other.
● Enemy units enter the playing field from the top of the map and move in the general direction of the player’s base.
● The player can build units from anywhere, but units are built at the player’s base.
● In order to build a unit, the player must select the unit’s mech type and the AI module that will control the unit.
● Units engage in combat with physical weapons such as shotguns, machine guns, rocket launchers, etc.

7.2 Movement
All units present in the physical move can move freely about the environment. Units are not constrained to any predefined pathways, but cannot pass through obstructions on the map (see Section 3.4) or each other. Automated units move based on the behavior they are running (see Chapter 9), while the player directly controls the movement of the hero unit.

7.3 Enemies in the Physical World
Enemy units will appear on the top of the map and travel toward the player’s base at the bottom of the map. Enemy units will have to contend with environmental obstacles, the player’s Hero, and the player’s units to reach their goal. Once enemies come within range of the player’s base, they will automatically begin to open fire on the base. New enemies appear in the physical world as defined by the current wave data, but also when an enemy manages to escape from cyberspace (see Section 8.3).

7.4 Building Units in the Physical World
In addition to the hero unit, the player is able to build autonomous units to aid the hero in defending the base. A button press brings up a radial menu (see Section 13.2) from which the player can choose the mech type and AI module of the unit to be built. Once these are confirmed, construction of the unit begins (assuming the player has enough RAM to build the unit), or the unit is stored in a queue if all of the player’s construction slots are full. Newly constructed units appear at the player’s base and begin to carry out actions according to their behaviors.

7.5 Physical World Unit Combat
Units in the physical world engage in combat with each other using conventional weapons such as shotguns, machine guns, and rocket launchers (see Section 5.3 for unit loadouts). How well a unit can aim depends on the behavior the unit’s AI is running, how fast the unit is moving, the weapon it is using, and the distance to its target. The hero’s aim is directly controlled by the player.
The health of units in the physical world is conveyed through visual effects such as scorch marks, sparking electronics, and machinery fires. Units have a single health attribute and are destroyed when their health reaches zero. Friendly units (including the hero) can seek repairs at a special area in front of the player’s base.
8. Cyberspace Gameplay

8.1 Cyberspace Gameplay Overview

**Note:** Firewall’s two playing fields are completely separate from one another. Therefore, all references to the “hero” or “enemies” in this section refer to the *cyberspace* version of the hero and enemy AIs present in *cyberspace*. It is important to remember that these are separate from the physical hero and physical enemy mechs, and do not interact directly with any units present in the physical world.

- When the player transitions into cyberspace, the cyberspace hero appears at the node closest to the hero mech’s location. The hero then becomes active in cyberspace.
- When an enemy mech is killed in the physical world, the AI controlling it is beamed to the nearest node in cyberspace. The enemy appears in cyberspace at this node and begins its escape attempt.
- The leaves of the cyberspace enemies’ behavior trees are represented as pods attached to the bodies of the enemies. In cyberspace, enemies are damaged and destroyed through the damaging and destruction of these pods.
- Destroying a behavior pod prevents the enemy that the pod belonged to from using that behavior. The inability to use behaviors linked to destroyed pods persists even after the enemy returns to the physical world should it manage to escape cyberspace.
- The player can build towers that orbit around a particular node. Towers attack enemies that are near the node on which the tower is placed.
- The MVP contains two tower types: one that attacks a single enemy directly, and another that has an area-of-effect pulse attack.
- All cyberspace units (hero and enemy) have beam weapons that they use to engage in combat with each other.
- When the hero is located at a node, a firewall can be deployed surrounding that node. Firewalls impede the movement of enemies and shield towers and the hero from harmful effects.
- As enemies approach the top of the map, malicious pulses are sent down and out from the top of the network to cover their retreat. These pulses damage the hero and any towers that they encounter as they spread through the network.

8.2 Movement

As mentioned in Section 3.3, unit movement in cyberspace is restricted to the network of nodes and pathways that make up the playing field. Cyberspace enemies can pass through each other without any negative effects, though they will attempt to avoid traveling on top of each other. They can also pass through the hero, and doing so deals a large amount of damage to the hero.

8.3 Transitions To and From Cyberspace

When the player transitions into cyberspace, the hero is projected into cyberspace at the node that is closest to the hero’s current location. Once the hero has materialized in cyberspace, the player takes control of it and can begin to move around and fight. When the hero transitions back into the physical world, the camera moves back to the focus on the hero mech, which becomes active in the physical world once again.

When an enemy mech is killed in the physical world, the AI that was in control of it is beamed to the cyberspace node nearest to the location at which the mech was killed. This transmission is conveyed through an effect that is visible in both the physical world and cyberspace. When the
effect is done playing, the enemy is revealed in cyberspace at the target node. The enemy then becomes active and begins to move about in cyberspace.

If an enemy reaches one of the exit points at the top of the cyberspace map, a visual effect is played to convey to the player that the enemy has been extracted from cyberspace and is being uploaded into a new mech. When the effect finishes, the enemy is no longer present in cyberspace, though it will return to cyberspace if its new mech is destroyed in the physical world.

8.4 Cyberspace Enemy Structure and Damage Mechanic

Enemies appear in cyberspace as a collection of pods attached to a central structure. Each pod is linked to a single leaf in the behavior tree of the enemy to which the pod belongs, and bears a symbol for that behavior to convey that link to the player. Rather than having a single health attribute that is depleted as the enemy takes damage, each enemy’s health is the sum of the current health of all of its behavior pods. Damage from attacks that hit the center of the enemy is distributed evenly across all of its remaining pods, but pods can also take damage and be destroyed individually. When all of an enemy’s behavior pods are destroyed, it is defeated and the rest of its body disintegrates.

The destruction of a behavior pod prevents the enemy that owned that pod from using the linked behavior again, even if that enemy should manage to escape back to the physical world. This allows the player to cripple enemy AIs in cyberspace, limiting their effectiveness in both planes. As mentioned above, pods are marked with colored symbols to communicate which behavior they are linked to, and pods that are linked to behaviors in the same branch of the behavior tree are grouped together. When all of the pods in a branch of the behavior tree have been destroyed, the AI is cut off from using that branch completely.

8.5. Building in Cyberspace

Instead of building automated units in cyberspace, the player has the ability to place towers at cyberspace nodes. These towers orbit around the node at which they are placed, and attack any enemies that come nearby. The player can build towers anywhere in cyberspace, regardless of the current location of the hero, though the hero must be in cyberspace for towers to be built. When the player presses the build button, control of the hero is relinquished and the player can pan around the map to choose the node at which the tower is to be built. Control of the hero returns once the location for the tower is chosen or the build process is cancelled.

8.6 Tower Types

In the MVP, there are two types of towers available for the player to build. The first is a direct attack tower that can target one enemy at a time. The tower fires a beam of disruption energy directly at that enemy, damaging whatever behaviors it hits (or doing some damage to all behaviors if it hits the enemy’s center). This tower type can effectively destroy individual behaviors quickly, since all of its damage is concentrated on whatever behavior pod its beam hits.

The second tower type that is part of the MVP attacks all enemies within a certain distance rather than targeting one enemy directly. When an enemy is within a certain distance of the tower’s node, it fires a pulse of energy into the node, which then travels outward along all of the pathways that connect to the node. Any enemies hit by the pulse take an equal amount of
damage to all of their behaviors. The pulse loses strength as it moves away from the node at which it originated, eventually disappearing completely. The tower continues to fire pulses at regular intervals until there are no enemies within range of its node.

8.7 Cyberspace Unit Combat

Both the cyberspace hero and the cyberspace enemies have beam weapons that facilitate combat in cyberspace. These weapons can be fired across the space in between the nodes and pathways, but have limited range (i.e. they cannot fire all the way across the map). The hero uses its weapon to attack the behavior pods attached to the body of an enemy, or to attack the center of the enemy itself, which damages all of that enemy’s behavior pods equally. Enemies can use their weapons to attack the hero or to attack any towers that may be within range (unless all of its fighting behaviors have been destroyed).

The player has direct control over the hero’s aim, which allows the player to strategically target specific behavior pods. For example, a player could take out all of an enemy’s Fight behaviors to prevent that enemy from fighting back, at the risk that the enemy might escape quickly using Self-Preservation behaviors. Alternatively, the player might take out an enemy’s Self-Preservation behaviors first to prevent it from escaping quickly (or escaping at all), though that enemy will then fight the player and the player’s towers much more fiercely. (See the AI Behaviors section for more information on behavior categories and how they work).

In addition to the beam weapon, the cyberspace hero unit has the ability to deploy firewalls around nodes. The hero can only deploy a firewall around a node if the hero is located at that node. Firewalls surround nodes and function in the following way:

- When the firewall’s health is very high, enemies cannot pass through it at all and take a large amount of damage if they collide with it. The firewall completely blocks any hostile attacks that attempt to pass through it.
- When the firewall’s health is moderate, enemies can pass through it, but do so slowly and take damage as they pass through. The firewall partially blocks any hostile attacks that attempt to pass through it.
- When the firewall’s health is low, enemies can pass through it without taking damage, but are slowed as they pass through. The firewall does very little to block hostile attacks.

Firewalls are damaged each time an enemy or hostile attack collides with them, but are unaffected by the movements or attacks of the hero and the player’s towers.

As enemies approach their escape points of the map, pulse attacks emanate from the exit points at the top of the map to cover their retreat. These pulses are similar to the attacks of the area-of-effect towers, but are much larger and more powerful. The pulses damage any towers built on nodes that they pass through, as well as do damage to the hero. The damage done by the pulses decreases over distance. The primary purpose of this mechanic is to prevent the player from building exclusively at the top of the map, which would make preventing enemies from escaping fairly easy. Instead, towers built at the top of the map will be destroyed by pulse attacks after several enemies come close enough to trigger the pulses, forcing the player to have a balanced building strategy. The pulses also make camping at the top of the map fairly dangerous for the hero.
9. AI System

9.1 AI System Overview
- An individual AI entity in our system is called an actor. Each game entity not under the control of the player (i.e. every unit except the hero, plus every tower) has a unique actor associated with it.
- Actors maintain their own internal state as well as data representing the actor’s perception of the game world.
- The basic actor update cycle will be as follows:
  - Actor updates internal state data and perceived world state data.
  - Actor’s behavior is updated, producing output dictating actor’s next action.
  - Actor carries out actions based on behavior output.
- The behavior of actors is dictated by a behavior tree structure, which facilitates behavior choice and execution.
- In addition to their persistent state and perception data, actors maintain temporary state data for the behavior that is currently being executed.
- For the MVP, three behavior categories (non-leaf behaviors) will be implemented: Idle, Fight, and Self-Preserve.
- The MVP children of the Idle behavior are Stand Ready, Advance, Patrol, and Guard.
- The MVP children of the Fight behavior are Attack, Pursue, Strafe, Flank, and Special Fight (which is replaced by the Snipe, Missile Barrage, or Cruise Missile behavior depending on unit type).
- The MVP children of the Self-Preserve behavior are Flee, Tactical Retreat, and Seek Repairs.
- The playing field in the physical world will be divided into a grid for pathfinding purposes.
- The playing field in the digital world will be represented as a graph for pathfinding purposes.
- Pathfinding will be done using the A* search algorithm.
- The cost to move to a new node during pathfinding will be based not only on distance, but on other dynamic factors such as the location of friendly and hostile actors, line of sight to the actor’s target, etc.

9.2 Actors
Actors are the basic AI entities of Firewall. Each unit not under the player’s control (including towers) has a unique actor assigned to it that controls how the unit behaves during gameplay. Actors maintain three important sets of data: the actor’s own state, the actor’s perception of the world around it, and temporary state data associated with the behavior the actor is currently running.

Things that constitute actor’s state are pieces of information such as position, velocity, aiming direction, health, perceived danger level, etc. The actor’s perception of the world around it is a list of “perceived objects,” data structures that represent one other entity that the actor is aware of. It is important to note that the state of a perceived object as maintained by an actor might not match the actual state of that object in the game world. The nature of behavior state data is dependent on the behavior that the actor is running, and does not persist when the actor changes behaviors (i.e. it is allocated and deallocated with each behavior change).

The actor update cycle consists of three main steps:
1. The actor updates the elements of its internal state that do not rely on exterior factors (i.e. position, velocity, etc.). Then, the actor updates its perception data. Once this is
done, the actor fills in the remaining state data that depends on perception data (i.e. danger level).

2. The actor updates its behavior. This involves deciding which behavior to run and then running the chosen behavior (see the next section for more information about how behaviors work). The output of the behavior update is a set of instructions that dictate what actions the actor should take next.

3. Based on the output of the behavior update, the actor takes actions in the game world. This step may not involve much if the actor is already in the middle of an action or the behavior update did not yield any new actions for the actor to begin.

9.3 Behavior Tree Architecture

The different behaviors that an actor can run are separated out into discrete classes which are then organized into a tree structure. Each behavior has several essential functions:

- **Evaluate Desirability**: Is this behavior desirable given the actor’s current state and surroundings?
- **Activate**: Called when a behavior first starts execution. Sets up temporary state data for the behavior.
- **Update**: Perform the actions associated with the behavior. Continuously called as long as the behavior is running.
- **Deactivate**: Called when a behavior ends execution. Cleans up temporary state data for the behavior.

In addition to calculating the desirability of a behavior, the system must have a way of checking whether or not it is even possible for an actor to run a behavior given the actor’s current circumstances. We will call this condition “validity.” The conditions that determine behavior validity are all binary, and the current value of those conditions are determined by an actor when it updates its internal state. The condition value set required for a behavior to evaluated will be defined in data and stored as a bitvector at load time. When considering behaviors for validity, this bitvector will be compared to the bitvector constructed by the actor when it updated its state.

Behaviors that occupy non-leaf positions in the tree do not actually perform actions, but make decisions about which of their child behaviors should be chosen for execution. The children of non-leaf behaviors are organized by priority, i.e. the highest priority child is first in the list and the lowest priority child is last. Leaf behaviors encapsulate the actions that the actor can take. The process for choosing and running a behavior is as follows:

1. Starting with the root (empty) behavior, each child behavior is considered for validity and desirability, in order of priority. The first behavior that is both valid and desirable is chosen, and execution moves to consider that behavior’s children. The first leaf behavior found to be both valid and desirable is chosen for execution.
2. If the chosen behavior is different from the behavior that the actor is currently running, the old behavior’s Deactivate function is called and the new behavior’s Activate function is called.
3. The current behavior is updated, producing output that will control the actions of the actor.
The structure of the behavior tree for the MVP is as follows:

![Behavior Tree Diagram]

**9.4 MVP Behaviors**

For the MVP, the following behaviors will be implemented. Leaf behaviors are grouped under their parent non-leaf behavior. All behaviors are listed in order of priority, with higher priority behaviors appearing first. In the behavior descriptions, “hostile” refers to an actor that is not on the same team as the actor running the behavior, while “friendly” refers to an actor that is on the actor’s team. “Enemy” refers to an actor that is hostile to the player, while “player-built” refers to an actor that is friendly to the player.

**Self-Preserve**

Self-Preserve behaviors are defensive in nature and are chosen when an actor perceives that it is damaged or in danger of being destroyed. The MVP Self-Preserve behaviors are as follows:

**Self-Repair**

The actor seeks to heal damage it has suffered during battle. This behavior functions differently for player-built actors than it does for enemy actors. A player-built actor (which always controls a unit in the physical world) running this behavior will flee to the player’s base, which has a special repair bay that will heal the actor upon its arrival. The behavior terminates when the actor is fully healed or when it is healthy enough that Self-Repair is no longer its highest priority.
Enemy actors can only use Self-Repair while in cyberspace. An enemy actor running the Self-Repair behavior seeks an empty node away from danger and, upon its arrival, begins rebuilding or repairing behavior pods that have been damaged or destroyed. The time required to rebuild a behavior pod is proportional to the complexity of the behavior linked to the pod. The behavior terminates when all of the actor’s behavior pods have been rebuilt or when the actor becomes endangered enough that running Self-Repair is no longer desirable.

Self-Repair can only be triggered when the actor is not under immediate threat from hostile forces. This could occur outside of combat (i.e. no hostiles are in the area) or potentially after another Self-Preserve behavior has moved the actor out of a major combat zone. Once the behavior becomes active, the actor takes an extremely cautious route to reach its repair location, and does not engage hostiles along the way.

**Tactical Retreat**
The actor moves away from hostiles that are threatening it, intermittently firing back at pursuing hostiles. The shots fired during the execution of this behavior are not as accurate as those fired during the Fight behaviors, and are meant more to discourage and hinder pursuit than to do lethal damage. While the actor’s primary goal is to move away from hostiles, it also attempts to move toward friendly units that may be able to rescue it from danger.

Tactical Retreat is only triggered when the actor is aware of hostile actors. The primary factor for triggering Tactical Retreat is the actor’s perception that confronting or continuing to confront the hostile forces directly is too dangerous. However, the actor must also be aware of nearby friendly forces (or the recent proximity of friendly forces) in order to begin executing Tactical Retreat. The behavior continues to run until the actor is no longer being threatened, becomes cornered, or after a certain amount of time.

**Flee**
The actor moves away from hostile actors and dangerous areas. The actor does not fire at hostiles while executing Flee, allowing it to move at maximum speed. The only factor in determining the actor’s movement preference is the location of hostile actors and dangerous areas (i.e. the actor does not take into account the location of friendly actors when determining movement).

Flee is only triggered when the actor is aware of hostile actors. If the actor perceives that engaging or continuing in combat with the hostiles that are threatening it is too dangerous (and it is unable to use Tactical Retreat), it use Flee in an attempt to escape those hostiles. The behaviors continues to run until the actor is no longer being threatened, becomes cornered, or after a certain amount of time.

**Fight**
Fight behaviors are offensive behaviors that are executed during combat with other actors. All of the Fight behaviors can only be triggered if the actor is aware of at least one hostile actor. The MVP Fight behaviors are as follows:

**Special Fight**
The Special Fight is not itself an executable behavior but rather encompasses a set of behaviors that are each specific to one of Firewall’s unit types. These behaviors allow each unit to make use of a special weapon that it does not have access to when running
its normal Fight behaviors. For the purposes of simplifying the AI customization process, the unit-specific Special Fight behaviors are not exposed to the player, since the one that is chosen depends entirely on the type of unit the actor is assigned to. When an actor that has access to the Special Fight behavior is assigned a unit, the proper behavior is inserted into that actor’s behavior tree in the place occupied by Special Fight. The Special Fight behaviors and the unit types they correspond to are as follows:

**Snipe (Scout)**
The scout keeps its distance from hostiles and attacks them with its sniper rifle. Snipe is triggered when no hostiles are immediately threatening the actor and when all possible targets are outside the sniper rifle’s minimum effective range. The target chosen is the most dangerous hostile in range. The actor will attempt to move to a location that has a clear line of sight to the target, is out of the way of the target’s aiming vector, and is close to the sniper rifle’s optimum range from the target.

Once a reasonably suitable location is found, the actor will remain at that location and fire at the target until the target is killed or the actor’s location becomes unsuitable for attacking the target with the sniper rifle. When this happens, the actor will first attempt to find a new target to Snipe, then attempt to move to a new location that is favorable for attacking its old target. Snipe continues until no more targets remain or a hostile moves inside the sniper rifle’s minimum effective range and attacks the actor.

**Missile Barrage (Infantry)**
The infantry unit unleashes a devastating missile barrage on a group of enemies. Missile barrage is triggered when the actor perceives a group of hostiles that are clumped together. The hostiles must be a safe distance away from any friendly units and outside the minimum effective range of the infantry’s missile launchers. The actor will attempt to move to a location that has a clear line of sight to most of the hostiles in the group and is close to the missile launchers’ optimum range from the group of targets.

Once the actor reaches a suitable location, it launches a barrage of missiles at the group of hostiles, targeting an area that it believes will encompass as much of the group as possible at the time of the missiles’ arrival. After the missiles are fired, the behavior terminates, and cannot be run again until the launchers have reloaded. Missile Barrage can also be interrupted if the actor is attacked while it is moving to the launch location.

**Cruise Missile (Artillery)**
The artillery unit fires a guided cruise missile at a single target, inflicting massive damage. Cruise Missile is triggered when the actor perceives a particularly dangerous enemy at extremely long range. The actor cannot use Cruise Missile if it is under heavy attack, but light danger will not prevent the behavior from being triggered. If the actor is not stationary, it stops before beginning its attack. Once it is in place, it launches a guided missile at its target.

The missile launched by this behavior actively seeks out the target and inflicts a large amount of damage upon impact. However, the behavior itself terminates as soon as the missile is launched. It cannot be run again until the missile launcher...
reloads. Cruise Missile can be interrupted if the actor’s line of sight to the target becomes obstructed by the environment before the missile is launched.

**Flank**
The actor maneuvers around its target to attack it from the sides and from behind. Flank is triggered when the actor engages a hostile that is not attacking the actor. The actor attempts to move to a position that is behind or to the side of a hostile unit (the hostile’s aiming vector is its forward vector in this case), provides a clear line of sight to the hostile, and is close to the actor’s optimum weapon range from the target. Once the actor reaches such a location, it will stay there as long as that location remains suitable, but will not hesitate to maneuver to a new location if conditions change (i.e. the target moves or starts aiming in a different direction).

The actor attacks the target continuously while executing Flank, provided that the target is within range. Flank continues to run until the target is destroyed, obstructed from view by the environment, or the actor’s situation becomes too dangerous for it to continue to focus on the target. Flank is aborted if the target’s aiming vector is pointing at or near the actor for a certain period of time (i.e. the actor is failing to maneuver behind the target).

**Strafe**
The actor attacks its target while moving around it. Strafe is triggered when the actor’s target is in range and is aiming at or near the actor. The actor moves toward the least dangerous location that is still in range of the target, provides a clear line of sight to the target, is on the same side of the target as the actor, and is out of the way of the target’s aim and direction of motion. The result of this should be that the actor moves in a direction relatively perpendicular to the target’s aiming vector. When the actor reaches the chosen location, it will choose a new one if the target’s aim has continued to follow it.

The actor attacks the target continuously while executing Strafe, provided that the target remains within range and a clear line of sight is maintained. Strafe continues to run until the target is destroyed, obstructed from view by the environment, or the actor’s situation becomes too dangerous for it to continue to focus on the target. Strafe is aborted if the actor reaches its current target location and the target is no longer aiming at or near it.

**Pursue**
The actor moves to intercept its target. Pursue is triggered when the actor’s target is near or beyond the actor’s maximum range. The actor attempts to move to a position that will be near the actor’s optimum range from the target when the actor arrives there. As long as the actor is within range of the target, it will attack the target while Pursue is running. Pursue continues to execute until the target is at or near the actor’s optimum range, the target is destroyed, or the actor’s situation becomes too dangerous for it to continue to focus on the target.

If the target becomes obstructed from view by the environment, Pursue will continue to run based on the target’s last known movement direction and speed. If the actor cannot perceive the target upon reaching the desired location, Pursue is terminated and the actor switches to a different behavior.

**Attack**
The actor attacks its target. Attack is triggered only if no other Fight behaviors are appropriate or available. If the actor is moving, it slows to a stop to achieve maximum
accuracy. The actor will only move if its target comes within its minimum effective range, in which case the actor will attempt to move away from the target until the target is once again at the proper range. Attack continues to run until the target is destroyed, out of the actor’s weapon range, or no longer perceivable by the actor.

**Idle**

Idle behaviors are non-combat behaviors that are executed when the actor is not aware of any hostiles, or is for some reason unable to run any of its Self-Preserve or Fight behaviors.

**Guard**

The actor moves toward an object that it would like to protect and attempts to remain close to that object. Guard works differently for player-built actors than it does for enemy actors. Player-built actors will move to the player’s base while running Guard, or to the hero if the actor can perceive the hero and it is closer to the hero than the player’s base. If the hero is chosen as the Guard object, Guard continues to run until the actor perceives an enemy. If the player’s base is chosen, the area around the base is designated as the actor’s assigned area and a flag is set indicating that the actor must remain in that area. In this case, Guard terminates once the actor is in the area around the base or if an enemy is perceived on the way to the base.

Enemy actors will only ever run Guard if their Fight behaviors have all been destroyed (see the section on Cyberspace Gameplay for an explanation of how this can happen), since enemies are offensive in nature and therefore prefer moving toward the player’s base than guarding anything. However, if an enemy actor is unable to use any of its Fight behaviors, it may use Guard during combat. Guard is triggered if the actor is aware of a friendly that is under attack from a hostile. The actor will attempt to move to a location that obstructs the hostile’s attack vector to the friendly, effectively acting as a shield for the friendly. Guard continues to run until the friendly is no longer in danger or the actor is destroyed.

**Patrol**

The actor moves around within a constrained area. The actor first chooses an area to patrol, then moves about in a pattern that evenly covers that area. The patrol area can be chosen in one of two ways. The first is by Guard, which can designate the area around the player’s base as the patrol area. If this was not done, the patrol area is chosen based on where the actor last perceived hostile units and where friendly units are currently placed. The actor will generally patrol areas in which hostiles recently appeared and few friendlies are currently located. If the actor patrols an area for a certain amount of time without perceiving a hostile and the guard flag is not set (i.e. the actor is not required to remain in that area), Patrol will terminate and become unavailable for a brief period of time. Otherwise, Patrol continues to execute until the actor perceives an enemy.

**Advance**

The actor advances toward its objective. This behavior works slightly differently for player-built actors than it does for enemy actors. Player-built actors running Advance will move toward a location somewhere in the middle third of the map, as this is where most battles will take place. Enemy actors running Advance move toward the player’s base, as destroying it is their primary objective. Advance terminates when the actor reaches its destination or engages a hostile. Player-built actors will always terminate Advance in
favor of engaging hostiles, while enemy actors may continue to run Advance if the
hostile does not present much of a threat.

Stand Ready
The actor remains in one place, ready for action. This is the simplest behavior in the
entire tree and effectively amounts to the actor doing nothing. The actor will periodically
turn its head to survey the playing field, but other than that takes no actions. Stand
Ready never terminates on its own, and will continue to execute indefinitely if no higher-
priority behaviors are triggered.

9.5 Pathfinding
Both the physical world and cyberspace playing fields will be represented as graph structures
for the purpose of pathfinding. For the physical world, this graph will be built from a grid that
covers the entire playing field. Each square in the grid is a node in the graph, with connections
to each adjacent square (including diagonals). Because cyberspace is already made up of a
network of nodes and pathways, no further abstraction is required to obtain the pathfinding
representation; the graph is simply built to reflect the playing field directly.

Basic pathfinding will be done using the A* graph search algorithm. The heuristic used to
estimate the total cost from the current node in the search to the destination node will be
straight-line distance. Nodes can be marked as impassable if they are covered by some sort of
obstacle. In addition to this standard implementation of A* for route calculation, our pathfinding
system will be able to take into account a number of different factors when determining how an
actor will reach its goal. The purpose of these extra factors is to allow an actor to not simply find
the shortest path to its destination, but to find the path most appropriate to the behavior that the
actor is currently running.

The extra pathfinding factors that will be included in the MVP are as follows:
- Closest Friendly - The shortest distance from the node to a perceived friendly actor.
- Friendly Proximity - The average distance from the friendlies perceived by the actor to
  the node.
- Closest Hostile - The shortest distance from the node to a perceived hostile actor.
- Hostile Proximity - The average distance from the hostiles perceived by the actor to the
  node.
- Friendly Line of Fire - Whether or not and how many friendly actors have aim vectors
  passing through the node.
- Hostile Line of Fire - Whether or not and how many hostile actors have aim vectors
  passing through the node.
- Angle to Target Line of Fire - The angle between a vector from the node to the actor’s
  target and the target’s aim vector.
- Line of Sight to Target - Whether or not the line of sight from the node to the actor’s
  target is clear.
- Change of Direction - The difference between the direction from the actor to the node
  and the actor’s current movement direction.
- Recent Danger - Whether or not the actor has recently witnessed friendly deaths or
  weapon strikes at the node’s location.

When considering nodes for travel, these factors add a preference or avoidance (negative
preference) value that is considered alongside the regular distance heuristic. Factors will be
added, modified, and/or removed as needed to produce the appropriate movement patterns for each behavior. The extra factors that an actor pays attention to depends on the behavior that the actor is running. The factors that are considered for each behavior and the strength of their influence on the actor's pathfinding during the behavior's execution are defined in data. For efficiency purposes, extra factors may not be considered during the evaluation of every single node in a pathfinding calculation.
10. AI Customization

10.1 AI Customization Overview

- Every time the player builds an automated unit in the physical world, the player must assign a custom AI module to control that unit.
- Each custom AI module has access to some behaviors in the behavior tree, but not others. The player can define modules by specifying which behaviors are available and which are not.
- The definition of custom AI modules is facilitated by the AI customization menu, which is accessible in several different ways.
- Modules can be saved by the player for quick selection during gameplay, speeding up the unit building process.
- Each behavior in the behavior tree has a complexity value. If the total complexity of the behaviors available to an AI module exceeds a certain threshold, there is a chance that a unit controlled by that module will turn on the player.

10.2 Defining Custom AI Modules

Like the enemy units, the automated units that the player can build in the physical world are controlled by AIs that use a behavior tree architecture. However, unlike the AIs in control of the enemy units, the AIs in control of the player’s units do not have access to every behavior in the tree. Instead, the player can define custom AI modules. An AI module is a tree of behavior nodes. These nodes are the ones activated for the current module and represent a subset of all possible behaviors. Whenever the player builds an automated unit, one of these AI modules is chosen (or a new one is defined) to control that unit over the course of its lifetime. Multiple units can be controlled by the same type of AI module (i.e. those units have access to the same set of behaviors), but it is important to note that each unit is controlled by a unique instance of the AI module assigned to it.

AI customization takes place on a special menu screen that can be accessed in four ways:

1. From the main menu (saved AI modules are associated with player profiles).
2. Through a prompt that appears before each wave asking the player if he/she wants to define any custom AI modules before the wave starts.
3. From the pause menu.
4. Through a button press when the build menu is active during gameplay.

The centerpiece of the AI customization menu is a display of the AI behavior tree, with behaviors organized by category. From the AI customization menu, the player can define new custom AI modules as well as modify and delete modules that have previously been saved.

If the player chooses to create a new AI module, the AI behavior tree is presented to the player with only the most basic idle and attack behaviors activated. This is the simplest AI module possible, as each unit must have at least one behavior that it can run outside of combat as well as at least one behavior that is useful in a fight. From this starting point, the player can choose to turn on other behaviors by highlighting them in the behavior tree and then selecting them. When a behavior is highlighted, a summary of its function is presented to the player, along with a video that shows a unit running the behavior. Once the player has activated the desired combination of behaviors, a button press brings up a window that allows the player to name the new AI module, after which the module is saved to the player’s profile.
To modify a previously saved module, the player can select an option at the top of the screen that will bring up a window containing all of the modules that the player has saved in the past. From this window, the player can select an AI module to modify or delete modules that are no longer wanted. When the player selects an AI module to modify, the window disappears and the behavior tree display is updated to show the active and inactive behaviors for that module.Customization then proceeds as it would for a new module, with the player turning behaviors on and off by highlighting and then selecting them. When finished, the player can choose to either overwrite the old AI module with the new one or to save the new module under a different name, preserving the original.

10.3 AI Module Complexity Mechanic

Each behavior in Firewall’s AI behavior tree has a complexity value associated with it, which is a general measure of the behavior’s effectiveness in battle. The deadlier the behavior, the higher its complexity value. For each AI module defined by the player, the complexity values of all of the module’s available behaviors are totaled to produce the module’s overall complexity value. Firewall’s fictional theme presents the idea that overly complex AIs are able to achieve self-awareness, which inevitably results in those AIs turning on humanity. As such, the player must take care to keep the complexity values of custom AI modules in a safe range or risk having units controlled by that module change sides in the middle of battle.

AI modules with complexity values below a certain threshold are not at risk of achieving self-awareness, so units assigned these modules will never turn on the player. However, any custom AI module with a complexity value that exceeds this threshold may cause units it is assigned to turn on the player. The flow chart in Figure 10.1 illustrates the process by which an overly complex AI module achieves self-awareness and turns on the player.
Initially, units controlled by a risky AI module function normally, but each time the AI for that unit switches behaviors it has a chance of beginning to become self-aware. This chance is proportional to the amount by which the AI module’s complexity exceeds the self-awareness complexity threshold as well as the number of times the AI has switched behaviors.

Once the AI module of a unit has begun to become self-aware, it is inevitable that the unit will turn on the player, but this does not happen immediately. Instead, the unit’s AI progresses toward self-awareness as it continues to switch behaviors and witnesses its automated brethren perish on the battlefield. The amount of time that this process takes is also proportional to the amount by which the module’s complexity exceeds the self-awareness complexity threshold. The higher complexity, the quicker the module achieves self-awareness. When the unit’s AI achieves self-awareness, it immediately switches sides, gains access to the entire behavior tree, and functions just as any other enemy would. This also means that the unit’s AI will be beamed into cyberspace (where it functions just as any other cyberspace enemy would) if the unit itself is killed.
Because increasing complexity increases the risk that units controlled by a given AI module will turn on the player, the player must choose carefully which behaviors to activate for each custom AI module, as activating most or all of the behaviors for any one module will cause units controlled by that module to turn on the player almost instantly. In addition, the module’s complexity value also determines how much RAM is required to build a unit controlled by that module (see the Resource System section for more on RAM), making units controlled by complex AI modules more expensive to build. Balancing the risk and cost of the AI modules assigned to physical units is an important component of a player’s strategy and play style.
11. Control Scheme

11.1 Control Scheme Overview
When prototyping Firewall, we experimented with two types of controls: A direct control scheme, where the user moved around the map with the use of WASD keys and attacked with the mouse, and Real Time Strategy control scheme, where the user used the mouse to point and click, designating a destination or target for the hero which it would then move to automatically. During the RTS controlled demo we quickly determined that the users preferred to be able to control the hero directly as opposed to commanding him. Our team learned that players enjoyed being closely involved in the combat of the game. The majority of the play-testers felt more connected and enveloped in the game play when actually controlling the hero.

It was decided that providing players with explicit control over the hero was the best option for Firewall. After extensive play testing, we came to the conclusion that the game was meant to be played with a controller. The Xbox 360 controller was the best method of input for the game. With the 360 controller we could allow for easy, intuitive control of the hero and efficient creation of units with the use of radial menus in the user interface. The following is the proposed control scheme for the Xbox 360 controller.

Without being controller specific, the actions that the player will be able to perform in the game are as follows:

- Control and move the hero around in both the physical and cyber spaces. In cyberspace the hero will only be able to travel on the node paths described previously.
- Aim the heroes weapons.
- Pause the game.
- Open the unit customization menus and select mech type and AI behaviors.
- Queue up a mech that was customized previously.
- Cancel the mech that was last placed on the production queue.
- Switch between the physical and cyber spaces.
- Zoom in or out of the playing field.
- Select and utilize hero abilities.

11.2 Xbox 360 Controls

**Physical Space Gameplay Controls (Xbox 360)**

- **Left Analog Stick**: Hero movement / Unit menu navigation
- **Right Analog Stick**: Aim hero crosshair / AI menu navigation
- **Start Button**: Pause Game
- **A Button**: Queue production of selected unit
- **B Button**: Reset production selection
- **X Button**: Universal ability
- **Y Button**: Switch between physical and cyber spaces
- **Left Bumper**: Zoom?
- **Left Trigger**: Hero ability
- **Right Bumper**: Toggle mech/AI menu on/off
- **Right Trigger**: Hero weapon fire
• **Directional Pad:** Select hero ability

Cyberspace Gameplay Controls (Xbox 360)
• **Left Analog Stick:** Hero movement
• **Right Analog Stick:** Aim hero crosshair
• **Start Button:** Pause Game
• **A Button:** Place selected tower
• **Y Button:** Switch between physical and cyber space
• **Left Bumper:** Zoom?
• **Left Trigger:** Raise a firewall around node
• **Right Trigger:** Hero weapon fire
• **Directional Pad:** Select tower type

Main/Start/Pause Menu & Radial Menu Controls (Xbox 360)
• **Left Analog Stick:** Menu navigation
• **A Button:** Select (highlighted) menu option
• **B Button:** Go back (if applicable)

While firewall is best played with an Xbox 360 controller, we do understand that our PC audience might not have access to one. The following is a proposed control scheme for keyboard and mouse input.
11.3 Keyboard + Mouse Controls

Physical Space Gameplay Controls (Keyboard + Mouse)
- **W, A, S, D**: Hero movement
- **Mouse**: Move hero crosshair / menu navigation
- **Escape**: Pause Game
- **Q**: Queue production of selected unit
- **R**: Reset production selection
- **E**: Universal Ability
- **Tab**: Toggle mech/AI menu on/off
- **Shift**: Switch between physical and cyber spaces
- **1-4**: Select hero ability
- **Right Mouse Button**: Hero Ability
- **Left Mouse Button**: Hero weapon fire
- **Mouse Scroll Wheel**: Zoom?

Cyber Space Gameplay Controls (Keyboard + Mouse)
- **W, A, S, D**: Hero movement
- **Mouse**: Move hero crosshair
- **Escape**: Pause Game
- **E**: Raise a firewall around node
- **Shift**: Switch between physical and cyber spaces
- **1-4**: Select tower type
- **Right Mouse Button**: Place selected tower
- **Left Mouse Button**: Hero weapon fire
- **Mouse Scroll Wheel**: Zoom?

Main Menu & Radial Menu Controls (Keyboard + Mouse)
- **W, A, S, D, Arrow Buttons, Mouse**: Menu Navigation
- **Enter, Left Mouse Button**: Select (highlighted) menu option
- **Escape Button**: Go back
12. Tutorial

12.1 Tutorial Overview
The tutorial will be its own separate level which can be selected from the main menu. It will be highlighted for first time players in order to suggest that they use it. The purpose of the tutorial is to familiarize a new player with the mechanics and interface of the game. The tutorial consists of several segments which are designed to introduce the player to everything the player will need to know in order to play. Most UI elements are hidden until revealed in the appropriate segment. Each segment consists of challenges for the player to complete, and possibly extras for the player to refine his or her skills. Each segment assumes the player has learned all previous segments. These are the segments:

1. Welcome
2. Learn To Move
3. Learn To Fight
4. Digital World
   a. Movement
   b. Attack
5. Firewalls
6. Towers and Resources
7. Mechs
8. AI tree
9. Putting it All Together

The Tutorial can be accessed from the Main Menu. In addition, players that have not played the game before will be prompted to go through the tutorial before they start playing (Figure 12.1).

First time playing? Try the Tutorial!

Figure 12.1 Tutorial prompt
12.2. Welcome
The welcome is a short monologue that introduces the player to the world of Firewall. It will introduce the major themes and gameplay mechanics in a short description. Afterwards it will introduce the tutorial interface to the player, which includes:

- An outline of the different tutorial segments.
- The ability to skip the rest of this particular segment.
- The ability to skip to any segment of the tutorial.
- That finishing a segment consists of completing all challenges.
- That once a segment is completed the player can choose to either move to the next segment, repeat the current segment, or perform extras.

Challenges:
1. Continue to the next segment.
Extras: none
Revealed:
1. Tutorial UI elements:
   a. Skip button
   b. Tutorial menu (switch segment button)
   c. Next segment, repeat, and extras buttons

Welcome to FIREWALL
Firewall is a top-down action strategy game where the player fights the final chapter in the war between man and machine by battling mechs in the physical world and the artificial intelligences controlling those robots in cyberspace.
Tutorial Navigation

The tutorial is comprised of segments. Each segment has a set of challenges to complete. You can skip the current segment at any time by clicking the Skip button. You can click on the Segments button at any time to choose a different tutorial segment. When all challenges have been completed, the skip button changes to a Continue button. Extra options may become available.

Challenge: Continue to the next segment.

Figure 12.3 Tutorial navigation instructions
12.3 Learn To Move
This segment introduces the player to the directly controllable hero unit. The player will be introduced to movement in the physical world.

Challenges:
1. Perform a series of movement tasks, such as traverse a triangle of points.

Extras:
1. Keep moving around.

Revealed: none

Figure 12.4 Hero movement instructions
12.4 Learn To Fight
In this segment, the player learns how to use the hero’s physical world attacks. The hero will also be introduced to the concept of enemy units. The player will have to use each of his 3 attacks on three separate enemies sequentially. The player will also be informed about his health and taking damage after the first enemy.

Challenges:
1. Defeat enemy 1 using primary attack.
2. Defeat enemy 2 using secondary attack.
3. Defeat enemy 3 using tertiary attack.

Extras:
1. Keep fighting an endless wave of enemies.

Revealed:
1. Health bars
2. Attack powers UI elements

Figure 12.5 Hero primary weapon instructions
Figure 12.6 Hero special ability instructions 1

Use secondary attack [Plasma Cannon]

Make sure you watch your health bar

Figure 12.7 Hero special ability instructions 2

Use tertiary attack [Rocket Barrage]
12.5 Cyberspace
Having defeated some enemies, the player will now be introduced to the cyberspace aspect of Firewall. The player will be introduced to movement between the two worlds.

Challenges:
1. Move to cyberspace.
Extras: none
Revealed:
1. World transition UI

Figure 12.8 Prompt to transition to cyberspace
12.6 Cyberspace Movement

The player will again learn movement controls, this time for cyberspace. The player will learn about pathways and nodes. (The pathway system will be very simple for the purpose of the tutorial.)

Challenges:

1. Perform a series of movements by following along a path.

Extras: none
Revealed: none

Figure 12.9 Cyberspace movement instructions
12.7 Cyberspace Attacks
The player will be introduced to the hero unit’s attacking capabilities in cyberspace. The player will also be introduced to the concept of health in cyberspace.

Challenges:
1. Destroy an AI

Extras:
1. Keep destroying AIs

Revealed:
1. Digital health UI elements

Figure 12.10 Cyberspace attack instructions
12.8 Firewalls
The player will be introduced to the placing of firewalls and the mechanics of slowing and diverting the escaping AIs.

Challenges:
1. Divert AIs to a choke point using firewalls.

Extras:
1. Keep destroying AIs

Revealed: none

Figure 12.11 Firewall placement instructions
12.9 Towers and Resources
In this segment the player will learn about the two cyberspace tower types. The player will also learn about the resources used in the game.

Challenges:
1. Place primary tower type
2. Place secondary tower type
3. Kill 3 enemies using only towers and firewalls

Extras:
1. Keep destroying AIs

Revealed:
1. Digital resource UI elements

Figure 12.12 Cyberspace tower building instructions
12.10 Mechs
In this segment the player will learn about creating his or her own mechs to aid against the enemy mechs. The three different mech types will be introduced, and the player will also learn about how both realms share the same resource system. The player can also learn about countering using mech types. The player uses preset AI modules during this section.

Challenges:
1. Create mech type 1
2. Create mech type 2
3. Create mech type 3
4. Win a small skirmish with player created mechs only

Extras:
1. Successfully counter enemy mech arrangements with player created mechs
2. Use the hero unit to fight alongside player created mechs

Revealed:
1. Mech creation UI elements
2. Physical World resource elements
12.11 AI Trees
The player will learn about the AI trees. The player will both learn how to apply AIs to the player's own mechs as well as save preset AIs. The player will also learn what the enemy AI representations in cyberspace actually mean (see Section 8.4). Finally, the player will learn about the restrictions on AI creation.

Challenges:
1. Create and save 3 AIs
2. Create player mechs with the saved AIs
3. Create an effective counter to an enemy mech
4. Create an AI which has too many nodes selected.

Extras:
1. Counter different wave makeups

Revealed:
1. AI selection and creation UI elements

Figure 12.14 AI Customization instructions
12.12 Putting it All Together
This final segment fully adds together the physical world and cyberspace. It also introduces the mini-map. Finally, the cyberspace pulse (see Section 8.7) will be introduced.

Challenges:
1. Defeat a wave of enemies in the Physical World
2. Fight them in the Digital World
3. Repeat until the AIs are fully destroyed and do not return
4. Start a game

Extras: none

Revealed:
1. Minimap (Physical and Digital)

Figure 12.15 Final overview of complete game
13. Screen Layout and Description

13.1 HUD Layout and Description
The image below illustrates all the possible menu’s and heads-up display elements a player will view during standard game play in both the physical world and cyberspace. Each elements title is followed by either a “P”, “C” or “P/C” in parenthesis. This tells us whether the element is in the physical world, cyberspace or both respectively.

Figure 13.1 HUD mockup
Figure 13.2 HUD concept

A description of each element, starting from the top left to the bottom right follows.

- **Hero Image:** This is simply an image of the hero, one for each world, that changes based on which world the player is in. This element is located in both the physical world and cyberspace.

- **Hero Health:** This an element that displays the current state of the hero’s health. This element is displayed across both of the worlds despite the different health systems for each plane. In the physical world the health represents the actual life of the hero. In cyberspace the health represents the players connection strength to the digital world.

- **RAM:** RAM is the amount of units and turrets the player can create. Essentially it serves as a maximum limit to keep the player from creating too many units and turrets. This element is displayed across both of the worlds since it is a shared resource.

- **Processing Power:** Processing Power is a resource that determines the speed at which the units and towers are created by the player. This HUD element is consistent across both worlds.
- **Unit Queue**: This element is used to keep the player informed of what order units and towers are being created in. The queue also limits how many units and/or towers the player can queue up for creation at once. This element is consistent across both the physical world and cyberspace to keep the player constantly informed of what is being created.

- **Mini-map**: This element is a scaled down version of the game map. In the physical world the mini-map is used to inform the player where the hero is on the field, where friendly units are and where enemy units are. When in cyberspace the mini-map will illustrate to the player where the hero is and where enemy units are. It will also share which nodes have firewalls and turrets on them. The minimap will show units in the opposite realm as a transparency much like how the game screen does.

- **Hero Ability Radial Menu**: This element is a radial menu that appears when the player is selecting a hero ability. It is located in the center of the screen and hides after about 2-3 seconds of inactivity. This menu can be displayed in both worlds however it’s contents will not necessarily be the same. Since the hero can have abilities specific to either the physical world, cyberspace, or even both, the abilities will be displayed in the menu respectively.

- **Unit Creation/ Tower Creation Radial Menu**: This element is a radial menu that will be used to create units and towers in the physical world and cyberspace respectively. This will exists in both worlds but is not consistent across both worlds. Since the player can only create units in the physical world the player will see the unit creation radial menu (see Section 13.3) but when in cyberspace the player will only see the tower creation menu, which follows the same schematic as the unit creation radial menu.

- **Artificial Intelligence Radial Menu**: This element is a radial menu that will allow a player to select from different AI Builds or create a new AI Build (see Section 13.2). This menu only exists in the physical world since the player can only create AI powered units in the physical world.
13.2 AI Selection Radial Menu Layout and Description

Layout Description
The AI selection menu will be displayed on the bottom-right hand side of the screen. The menu will contain four distinct elements. The elements on the top, left, and right of the menu will be AI builds that the player created or, if they haven’t created custom builds, preset builds that the game will provide. The element located on the bottom of the menu is called “Create New A.I.” and will allow the player to do exactly that, it will give the player the chance to build their own custom A.I. set. When this element is selected the player will be redirected to a new menu, that will pause the screen. For further detail please look at the “A.I. Customization Menu and Layout Description” section.

Navigation
- **Right Bumper:** Display the menu while the bumper is held.
- **Right Stick:** Allows the player to move among the different selectable elements
13.3 Unit Selection Radial Menu Layout and Description

Layout

![Unit Radial Menu Scheme 1]

**Description**
The unit selection menu will be displayed on the bottom-left hand side of the screen. The menu will contain three distinct elements. The elements on the top, left and right of the menu will be one of the three distinct units that the player can create.

**Navigation**
- **Right Bumper**: Display the menu while the bumper is held.
- **Left Stick**: Allows the player to move among the different selectable elements
13.4 AI Customization Menu Layout and Description
Seeing that one of the core mechanics of Firewall is to provide players with the ability to construct their own artificial intelligence it was only natural to have an interface that would allow the player to select specific AI behaviors. Below is a design of what the menu would look like.

Layout

![AI Customization Menu](image)

Figure 13.5 AI Customization menu

Layout Description
The AI customization is done via a radial menu that will temporarily pause the game and appear at the center of the screen. The menu is presented in the form of a curved triangle. Each side of the triangle represents one of the three distinct categories of the behavior tree: Fight, Self-Preservation and Idle. Take notice that each category has been divided into a unique number of subsections. These subsections will house the appropriate leaves of the AI behavior tree. These leaves will be represented by symbols. In the center of the menu there will be the name of the behavior followed by information of what exactly the behavior does, represented in text or through a quick video and, the complexity level of the behavior. At the top-right of this menu screen there will be an element that displays the level of complexity of the overall AI build. This will inform the player how complex the current build is and exactly how likely it is that the build will cause his or her units will turn on them.
For clarification it is important to know that this a sub menu that the player will go to when they chose the “Create New AI” option. The player can edit existing builds or create new ones. Both options will take you to the same menu, however, existing builds will have the appropriate active behaviors highlighted.

Navigation
A list of controls to navigate through the menu is described below.

- **Right Stick:** Allows the player to navigate through the different options in the menu.
- **“A” Button:** Toggle the behavior that is currently selected.
- **“B” Button:** Navigate back to the “Edit or Create” menu.
- **“X” Button:** Export the AI build.
- **“Y” Button:** Clear the AI Build.
14. Tools

14.1 Balancing Tool
Balancing is an important part of any game, and the purpose of this tool is to make the process of turning the knobs and dials of balancing as simple as possible. The tool will be developed in Adobe Flex SDK, which makes layout and getting that first screen up and running all but trivial, and has great XML parsing libraries. The most important reason why the tool will be written in Flex is that by using Flex SDK, the tool can be a standalone AIR app, and can be run without the need for Visual Studio to re-compile Firewall whenever a change is made. The tool will mostly be used by our development team, but can also be easily explained to non-developers during playtesting.

![Balance Tool Mockup - Unit Editor](image)

**Figure 14.1 Unit editor**

14.1.1 Unit Editor
The unit editor is for fine tuning the NPC units on the fly without having to recompile the game every time. The unit editor panel will have all the attributes available for manipulation on the loaded NPC via manually typing in values or adjusting sliders and knobs of parameters.

Explanation of Design:
(1) The title will go here, it’s simply a title.
(2) This is an accordion title, along with (11). The basic premise of an accordion is you click the bar for the page you want to go to, and it collapses all the others to show the one you chose. It is the same functionality as Tabs, but accordions keep clutter down and fully distinguish between the elements. The Unit Editor is designed for editing unit (NPC) parameters.

(3) This is the placeholder text for the NPC name. This would be “Scout”, for example.

(4) An image for the idle sprite. This is just to give the user an image of what they’re working with. This could possibly animate as well, depending on how the initial feedback goes.

(5) This is where the file information will be. Ideally, this would display who edited it last and when. This data may be retrievable from SVN, or it just may require users to manually author this.

(6) The Attributes list will display a clickable list (8) for all of the attributes in this object. This is probably the most complicated part of the tool, as it will maintain in memory a data object that holds all the attributes. It will make automatic backups (filename.tmp, for example) to prevent data loss, and does not get written out to the original file until you save it using the (18) save button. This prevents file conflicts and lets the user not worry about work being lost.

(7) This is the scroll bar, because the odds of the attributes list being long are high.

(8) The attributes will be listed in a numbered list, and when you double click on the attribute in the list, it will load that into the attribute viewer (13) window.

(9) Removes the currently selected attribute from the list.

(10) This adds a new blank attribute to the list.

(11) This is the behavior editor bar, press this to go to the Behavior Editor accordion view (the next image)

(12) The notification bar shows things like “This file is not saved yet” or “File successfully saved”.

(13) This is the Selected Attribute view. It displays all the information belonging to the selected attribute from the Attributes List (6)(8). The name is a human readable name, such as “Position”. It is not the variable in actual code.

(14) This is an editable text box for the comment on this attribute. Comments should fully explain the variable and how it’s used. This should include an object type (Vector2D, int, bool, etc)
(15) The variable name as it appears in code. It is important for debugging purposes to display it here, as well as when adding new attributes with (10).

(16) The value is what it’s value is set to. For example, if it’s a Vector2D, it could show “5,6”. For a boolean, it could show “false” or something.

(17) Saves your changes to the attribute into the temporary storage. This does not save it into the source file, however.

(18) Click this to select a file to open.

(19) Saves using the same file the user opened.

(20) Saves using the standard “Save as” dialog box, for saving experimental builds for instance.

**14.1.2 Behavior Editor**

![Behavior Editor Diagram]

Figure 14.2 Behavior editor

The Behavior Editor is going to be used to tweak influence values and other attributes specific to behaviors. The tool will function quite similarly to the Unit Editor, only it will specifically modify behaviors.
Explanation of Design
(21) The title of the tool.

(22) The Unit Editor panel collapsed. If you click this title, it will collapse the Behavior Editor and expand the Unit Editor again.

(23) The Behavior Editor panel title. This whole page is what shows after you’ve pressed the Behavior Editor title. It is one of the accordion views.

(24) The name of the currently loaded Behavior that the user has opened with (36)

(25) A brief description of the behavior in human readable format.

(26) The attributes list. This is the panel which displays all of the attributes of the selected behavior.

(27) The attributes are in a list here. If you double-click on the attribute, it will load the attribute for editing into the right window.

(28) Deletes the currently selected attribute. Will not crash when nothing is selected.

(29) Adds a new empty attribute to the list, and opens it in the right window.

(30) Notifications go here, such as “This file has not been saved” or “File saved as XXXX.xml”

(31) The opened attribute’s human readable name (not the variable in code).

(32) A brief description of what this variable is for and how it is used. There is also a description of the variable’s type (Boolean, Vector2D, etc)

(33) Variable name as it appears in code.

(34) The value for this variable

(35) Saves the attribute back up to the file stored in memory. This does not save to the disk, just to temporary memory.

(36) Opens a new “Open File” dialog box to choose which file to load.

(37) Writes the file to disk using the file the user chose in the “Open File” box.

(38) Opens a “Save As” dialog box to save the file into a different location.
14.1.3 Wave Editor
The Wave Editor tool will enable us to easily create and modify existing waves without having to dive through lines upon lines of XML. It will also offer us additional functionality that simply opening up an XML file would allow, such as automatically displaying numbers of specific units as well as giving an easily accessible summary of waves without manually parsing XML.

**Figure 14.3 Wave editor**

(40) The title for the Wave Editor.

(41) The name of the stage. Could be “tutorial” or such in a human readable format.

(42) A textual description of the wave.

(43) The panel that will list each of the waves included in this file.

(44) A numeric list of each wave, with a simple summary as the title. When you click on one of the waves here, it will load the selected wave into the detailed wave editor panel (49)

(45) Removes the selected wave from the list of waves
(46) Adds a new wave at the end of the list and opens it in the detailed wave editor panel (49).

(47) The notification bar, where things such as “File saved” or “Wave not saved yet” will appear.

(48) Displays the wave number as the title of this loaded wave.

(49) A detailed list of the units selected for this wave.

(50) Saves the currently selected wave into memory.

(51) Opens a new “Open File” dialog box, for opening a wave plan.

(52) Saves the currently loaded wave plan to the original file which was opened.

(53) Saves the currently loaded wave plan to a new file, which will be presented with the familiar “Save as” dialog box.

14.2 Artist Tool

14.2.1 Artist Tool Overview

For the artist spriting tool, Firewall will be using Adobe Photoshop to create sprite renderings and animations. We will be using the animation workflow introduced in Adobe Photoshop CS3 Extended. Once the artist has finished their animation, they will export the animation frames and then combine them into a spritesheet using an in-house software program.

14.2.2 Workflow

1. The artist will first draw their sprite in Photoshop, while doing so they must consciously separate parts of the sprite that will be animated into their respective layers and groups. The dimensions of the sprite must be in powers of 2 (e.g., 64x128, 256x256 etc...). Once the initial drawing is complete, the artist can begin animating by starting the animation tool (In Adobe Photoshop, this tool can be enabled by using Window->Animation). Before beginning animation the artist should make sure to set the Document Settings in the animation(timeline) tool to the duration and frame rate they would like to animate at. When creating new frames, the artist needs to make sure the frame time delay values are consistent.

2. Once the animation is complete and ready to export, select File->Export->Render Video... in the Photoshop tool menu. The Render Video tool will pop-up, first select the folder that the image sequence will be exported to as well as the file’s base name. Under File Options select Image Sequence with type PNG. The number of digits only need be large enough so that every frame is addressable in that number space (e.g., 2 digits means that 100 frames can be rendered, 3 digits means that 1000 frames can be rendered). Size can be left to Document Size unless the artist intentionally wants to change the sprite size. Under Render Options the Alpha Channel should be set to Premultiplied with White in order to preserve transparencies. Set the Frame Rate to the reciprocal of frame time delay used in the animation(frames) tool (e.g., if the frame time delay was set to 0.2 seconds, which is 1/5 then the Frame Rate should be 5). Setting the Frame Rate this way makes sure that each frame will only get rendered out once and
will appear the same in the game as it did in Photoshop.

3. The image sequence will be combined into a single spritesheet (xnb format). The images must be provided in their own directory with a file specifying how many frames long the animation is and how many frames per second it is as well.
15. Art Assets

15.1 Game Info (HUD) Assets

Firewall’s game HUD will be active at all times while the player is active in the game. This means, as long as the player has control of the hero they should be able to see all of the HUD information and access the radial menus.

Minimum Viable Product Assets

- Hero Image: the hero image will be visible in both the physical world and cyberspace. The hero image should be a close-up portrait of the hero, a la the portraits in Starcraft or Dragon Age. For the MVP game, there will only be one hero portrait because there will only be one hero. There should be a frame around the hero portrait that looks kind of metallic but also grungy or dirty. The frame for this, and all of the HUD should remind the player of Starcraft, but dirtier.

- Hero Health: the health bar image will be visible in both the physical world and cyberspace. The health should be a single red bar since that’s the convention in
games for health. As the player loses health, the bar will scale back so that the empty space in the frame indicates the amount of health the hero has lost.

- Processing Power: the processing power bar will be visible in the physical world and cyberspace. This bar should be almost identical to the health bar, but instead of having a red image, the image will be yellow. As the player spends more of the hero’s allotted processing resource, there will be more empty space in the frame to indicate that processing power is no longer available to the player.

- RAM: the RAM bar will be visible in the physical world and cyberspace. The RAM bar should be very similar to both the health and processing bars, indicating how much RAM is left for the player to use. As with the previous two assets, any empty space in the frame should represent how many resources the player is already using. The RAM bar will be green.

- Unit Queue: the unit queue frame will be visible in both the physical world and cyberspace. This frame will display which units the hero is building and how close each unit is to being completed. For the MVP, this queue should only have the unit name in text and a green bar which starts as invisible when the unit has just started being built and progresses towards 100% when the unit is finished.

- Minimap: the minimap should be visible in both the physical world and cyberspace. The minimap should be a scaled down version of the background for the current level. Since the MVP only has one background, there will only need to be one background image for the physical world. There will also only need to be one background image for cyberspace. In either realm, the hero should be represented by a green dot. Enemy units should be represented as a red dot, and the hero’s units should be represented as white dots. The hero base, if it is on the minimap, should be a larger green square, but at relative size to what it is in the game level. In cyberspace, the enemy base should be represented as a larger red circle. This should also be to scale with its game level counterpart.

- Radial Menus: radial menus are detailed separately in the radial menu section below.

**Nice-to-Have Assets**

- Hero Image: the hero image will be visible in both the physical world and cyberspace. The hero image should be a close-up portrait of the hero, a la the portraits in Starcraft or Dragon Age. For the Nice-to-Haves, there will be multiple heroes, so multiple portraits will also be necessary. Additionally, there should be several different images based on how damaged the hero is. For instance: the normal hero image is for 100% health, and as the hero’s health drops lower and lower, the hero appears more bloodied and tired. The frame for this, and all of the HUD should remind the player of Starcraft, but dirtier.

- Hero Health: the health bar image will be visible in both the physical world and cyberspace. The health should be a bar that fades from green to red as the player gets closer to dying, since that’s the convention in games for health. As the player loses health, the bar will scale back so that the empty space in the frame indicates the amount of health the hero has lost and the health will slowly
become more red to indicate to the player the hero is more damaged. To make it more obvious when the hero takes damage, the health bar should glow slightly whenever the hero is taking damage.

- **Processing Power**: the processing power bar will be visible in the physical world and cyberspace. This bar should be almost identical to the health bar, but instead of having a red image, the image will be yellow. As the player spends more of the hero’s allotted processing resource, there will be more empty space in the frame to indicate that processing power is no longer available to the player. Again, as the player consumes or regains processing power, the image should glow slightly to indicate this change to the player.

- **RAM**: the RAM bar will be visible in the physical world and cyberspace. The RAM bar should be very similar to both the health and processing bars, indicating how much RAM is left for the player to use. As with the previous two assets, any empty space in the frame should represent how many resources the player is already using. The RAM bar will be green. As the player consumes or regains processing power, the image should glow slightly to indicate this change to the player.

- **Unit Queue**: the unit queue frame will be visible in both the physical world and cyberspace. This frame will display which units the hero is building and how close each unit is to being completed. For the Nice-to-Haves, this queue should have the unit name in text, a small version of the sprite of the unit to be created and a green bar which starts as invisible when the unit has just started being built and progresses towards 100% when the unit is finished. Instead of displaying a percentage already completed over the green bar, a time to be completed should be displayed there.

- **Minimap**: the minimap should be visible in both the physical world and cyberspace. The minimap should be a scaled down version of the background for the current level, both physical world and cyberspace. In either realm, the hero should be represented by a green dot. Enemy units should be represented as a red dot, and the hero’s units should be represented as white dots. The hero base, if it is on the minimap, should be a larger green square, but at relative size to what it is in the game level. In cyberspace, the enemy base should be represented as a larger red circle. This should also be to scale with its game level counterpart.

- **Radial Menus**: radial menus are detailed separately in the radial menu section below.

### 15.2 Radial Menu Assets

#### 15.2.1 A.I. Radial Menu
The A.I. Radial Menu is displayed at the players will. This particular menu will require many specific assets. A concept sketch of the menu is displayed below.

Figure 15.2 Radial menu concept

- An image that can be used to represent the selectable elements of the menu.
- 2-4 images that for the connections for each of the elements.
- A center piece that will be placed in the center of the menu and function as a pointer indicating which element the user has currently selected.
- An image that of the selectable elements in a highlighted state.
- 4 distinct symbols or texts to help the player differentiate between the options in each selectable element.
- The assets need to have a transparency of about 30% so that they are visible but allow for vision of the game.

15.2.2 Unit Radial Menu
This menu is activated at the players will and is used to allow the player to select one of the three units for building. The menu will follow the style illustrated in the “A.I. Radial Menu” section above. This menu requires the following assets.

- An image that can be used to represent the selectable elements of the menu.
- 2-4 images that for the connections for each of the elements.
- A center piece that will be placed in the center of the menu and function as a pointer indicating which element the user has currently selected.
- An image that of the selectable elements in a highlighted state.
- 3 distinct symbols or texts to help the player differentiate unit types.
The assets need to have a transparency of about 30% so that they are visible but allow for vision of the game.

15.2.3 Hero Abilities Radial Menu
This menu can be activated at the player’s will. This menu will allow the player to quickly switch between the abilities the hero has. This particular menu will have a similar artistic style as the previous sections but will be notably different as well. The requirements for the assets are listed below.

- An image that can be used to represent the selectable elements of the menu.
- 2-4 images that for the connections for each of the elements.
- An image that of the selectable elements in a highlighted state.
- 2-4 distinct symbols or texts to help the player differentiate between the different abilities.
- The assets need to have a transparency of about 30% so that they are visible but allow for vision of the game.

15.3 Title Screen Assets
The title screen is displayed after the starting credits (production company, distributor, etc.) of the game. The title screen will require several assets such as a game logo, production company or team logo, a background and several game options. Though the differences should be minimal, I will detail the differences between the minimum viable product title screen and nice-to-have title screen.

Minimum Viable Product
- **Game Logo**: the game logo should be prominent on the title screen so that players will know what game they’re playing. Ideally, the game logo will be centered on the screen and the largest and most prominent thing on the screen. For the minimum viable product, the game logo should not be animated.

- **Production Team Logo**: the production team logo should be very minor on the title screen. It should be smaller than the game options, but still be visible on the screen. A small icon in a bottom corner of the screen or a small version of the logo would be ideal. For this version of Firewall, the logo should not be animated.

- **Background**: the background of the title menu should convey the general atmosphere of the game to a new player. For instance: a scene of a destroyed city, one of the level background or a dramatic picture of the hero working with friendly units.

- **Game Options**: the game options shouldn’t be anything fancy for this version. There should be several options available to the player such as ‘continue game’, ‘new game’, and ‘settings’. There should be some indication such as the options turning a different color when selected or a glow effect to let the player know what option they have selected.

Nice-to-Haves
- **Game Logo**: the game logo should be prominent on the title screen so that players will know what game they’re playing. Ideally, the game logo will be centered on the screen and the largest and most prominent thing on the screen. The game logo should be animated in some way that makes sense for the logo design. For instance if the title
includes fire, the fire in the logo should be animated. If there’s nothing that really makes sense to be animated in the logo, then a dissolve effect for the title appearing on the screen.

- **Production Team Logo**: the production team logo should be very minor on the title screen. It should be smaller than the game options, but still be visible on the screen. A small icon in a bottom corner of the screen or a small version of the logo would be ideal. The logo should not be animated.

- **Background**: the background of the title menu should convey the general atmosphere of the game to a new player. For instance: a scene of a destroyed city, one of the level background or a dramatic picture of the hero working with friendly units. This background should be animated somehow, whether it be changing static backgrounds or having a short, loopable animation of some of the game sprites.

- **Game Options**: the game options should be fancy for this version. There should be several options available to the player such as ‘continue game’, ‘new game’, and ‘settings’. There should be some animation to indicate which option is being such as particle effects, some sort of animated arrows or reusable game sprite (perhaps the hero unit showing up in front of the selected option), or a glow effect to let the player know what option they have selected.

### 15.4 Environment Assets

#### Minimum Viable Product

Our minimum viable product only includes one distinct level in both the physical and digital world. Though, in either the MVP or the nice-to-have the digital world will use the same environment assets.

#### Physical World

- **Background**: The physical world will be created using a series of tiles. These tiles will be designed with re-usability in mind so in the future (i.e. nice-to-have) we can use these tiles to create new maps. The purpose of this background is to give the player a feeling of what kind of world they’re in as well as being able to see the boundaries of the map, as the entire background will be bigger than a single computer screen. The entire background is not homogeneous, in other words, it is not the exact same texture and color throughout. Just like the real world, there needs to be variation in the type of terrain. For instance, patches of dirt, debris and rocks will be scattered throughout the background. Potentially, a road that’s been worn into the grass where many people have traveled. The background itself will not be animated.

- **Player’s base**: the player’s base is an object which can be seen at the bottom of the screen. The base is bigger than the player unit, it needs to be significantly larger because it houses many facilities such as a repair station etc... This base is in a similar state as the rest of the world: crumbling, rusty, dirty and with holes. To sum it up: somewhere you wouldn’t feel safe living. The base has an entrance and/or exit where the player can deploy units from. As more damage is done to the player’s base, it will begin to flash red or some other color to indicate it is in danger. The base also indicates when it is being hit when it’s being attacked. The player’s base need not be animated. The player’s base, like all other objects in this world, should be viewed from a top-down point of view.
- **Walls:** walls are in the game to try and trap the player and also to enable the player to try and route AIs or even trap them, as well. Walls are destructible. Walls will give indication that they are taking damage to give the player feedback. Even though the view is top-down, it should be noted that walls are tall enough to block the line of sight/line of fire (see Glossary for definition) between enemy and ally units. Walls should look rather dingy, crumbly and overgrown. Walls are not animated.

- **Boulders:** large rocks or perhaps parts of mountains that might be on the field. Boulders also obstruct player's and enemy's paths, but not to the extent that walls are meant to restrict them. Boulders are unmovable and indestructible. The boulder looks like a rock that is larger than any unit so that it makes sense these units can't move the boulders. Boulders will not be animated.

- **Rubble:** rubble is the remains of buildings long since destroyed by the rogue AIs or battles fought between the two sides. Some parts of the building that has been destroyed will still be visible so the player knows what it is. Rubble is destructible, when it takes damage, it will give some indication that it is taking damage. Rubble that is severely damaged will give some indication that they are severely damaged, such as permanently being red-hued or flashing constantly instead of just when being hit. When rubble is destroyed, it disappears from the map. Rubble is drawn from a top-down perspective. Rubble is not to be animated.

**Cyberspace**

- **Background:** the background for the cyberspace is primarily black. However, there are also hints of chaos in the blackness. For instance, a noise function being animated in the background so it looks like static on a T.V. or animated dark grey swirls as part of the background.

- **Nodes:** the nodes in cyberspace will be electric blue or purple. They will evoke a TRON-like feeling. The nodes will have blue, glowing, concentric circles to indicate where they are and increase their visibility since they will still be low contrast. Nodes will be animated in that the colored circles should have a glow effect that pulses to make them more obvious.

- **Paths:** paths in the cyberspace are used to connect two nodes together. Paths need to be just as visible and distinctive as the nodes. Paths have a glow animation and evoke a feeling/aesthetic similar to that of TRON. It would be ideal to achieve this effect with particles using high-level shader language.

- **Enemy Base:** the enemy base is a larger node at the top of the map. This base looks exactly like a scaled up version of the regular nodes, but instead of the circles being blue, they are red. The enemy base has an animation of a pulse emitted in a circle out from the edge of the base. This can be animated either by using particles, or an artist-made animation. The pulse will gradually fade in opacity as it travels farther from the base so that the edge of the circle should always be less transparent than the center of the circle.
Nice-to-Have
In the nice-to-have version of our game, we would like to include a few extra art assets. For instance, we would like different backgrounds for different levels and more obstacles for the real world levels.

Physical World
- **Background**: In addition to the MVP backgrounds, there will be multiple backgrounds. The different backgrounds will be varied from one another. For instance, the backgrounds could get progressively more dark, with less grass and more of a destroyed city feel to show the player is getting closer to his goals of destroying the AIs. Parts of the background are animated, such as grass randomly blowing in the wind or an insect passing through the level occasionally.

- **Player's base**: In addition to the MVP behavior of the base, the base also indicates when it is being hit when it’s being attacked. Contrary to the MVP, the player’s base will be animated so that as it takes more damage, more and more holes appear in it to give extra indication that it is taking damage.

- **Walls**: In addition to what is in the MVP, several different kinds of walls will be implemented, such as wooden fences, chain link fences and stone walls.

- **Rubble**: In addition to the MVP rubble features, the rubble will quickly crumble as it takes more damage, until it collapses. The collapsing rubble should also have a short animation of a dust cloud before the rubble disappears. Just like walls, in the Nice-to-Have version, there should be multiple types of rubble for different buildings. For example: houses, skyscrapers and warehouses (though the warehouses should be easily distinguishable from the player’s base). All of these types of buildings should also have their own breaking down animations associated with them.
15.5 MVP Unit Assets

Required Art Assets (Physical)
The MVP physical world unit assets that will be required for Firewall are the three units types (Scout, Artillery, Infantry, each with separate animated parts.

The above concept art is for the fast and light “Scout” unit. Because our game will be top-down, this art is only for reference of the parts we need. The Scout will consist of four separate pieces, which will be drawn onto the unit at run-time.

The red part is the base of the unit, which will point the direction it is moving. This base will require an animation of the tires moving. The blue part is the turret of the Scout, and will only require a static image of the turret. The yellow part shows the Scout’s main guns, of which it has one on both sides. These guns will require a shooting animation for each, combined on the same animation. The pink part is the big, secondary cannon for the scout, which will require a firing animation on a larger scale than the main (yellow) guns.
The above concept art is for the basic “Infantry” unit. Again, it is only for reference as our game is from the top-down view.

The blue box highlights the Infantry’s base, which has a pair of tank-like treads which need to be animated. The yellow box highlights the Infantry’s turret, which can be static. The red box shows the Infantry’s main cannon. The Infantry has two of these cannons, one on each side. These cannons need to be animated to fire one after the other. The green box highlights the Infantry’s rocket launcher pod. This pod houses rockets, which needs to be animated to portray a rocket being launched.
The above concept art is for the “Artillery” unit. It is shown for reference. The blue box shows the base of the Artillery unit. Because it is from the top down perspective, there will not be much tread showing, but it still will show some motion on the treads. The red box shows the turret of the Artillery unit. The green box shows the guided missile bay for the Artillery unit. This bay will have an animation for firing a very large missile out of it. The yellow box shows the Artillery unit’s main cannon. This cannon will be animated to portray a large scale cannon, much larger than all the other units’ guns, firing.

Each of the above mentioned boxes portrays a different layer of the sprite which will make up each unit.

15.6 Cyberspace Unit Assets
Cyberspace will require several unique units for it. Assets are needed for both the player’s towers and the enemies’ forms in cyberspace. For the MVP, the player has two towers and the enemies have a basic digital form that doesn’t necessarily correspond to their types. The Nice-to-Have version includes more variation in the assets for the player towers and more variations in the enemy types, visually.

Minimum Viable Product
The MVP digital unit assets will include two tower types, sharing possible identical art assets and a single enemy asset for all enemy types.

Hero Units
For the MVP, the hero will be able to place two different tower types in cyberspace: a short-range tower that uses a pulse to attack near by enemies and a long range tower that focuses a disruption beam on one enemy in its range.

- **Pulse Tower**: this tower needs a static image which will be placed on a cyberspace node. This image should be slightly smaller than the node so that the node can still be seen under the tower once the tower is placed on it. This tower should not be animated, but should have an animation of the pulse attack. This attack animation should look like a blue circle that starts out at full (100%) opacity and fades as it travels away from the tower. This can either be done by using a static circle that fades into completely transparent (0% opacity) and is scaled up using code, or a short animation by an artist.
- **Long Range Tower**: this tower needs a static image that, for the MVP, is mostly the same as the pulse tower’s image. The long range tower and pulse tower should be easily distinguishable from each other and the easiest way to do this would be to simply change the colors from the pulse tower so that the player can easily and quickly differentiate between the two tower types. This tower should not be animated. Like the other tower, it should have an attack animation. This tower’s animation should be a simple, blue line that can be scaled so it reaches an enemy the tower is currently attacking. This line should also be distinguishable from the paths connecting nodes. This differentiation can either be from a different shade of blue or an extra glow effect on the line, but there shouldn’t be any player who confuses a path and a disruption beam.

**Enemy Units**
For the MVP, enemy units will have a few static images and those images will likely all look the same despite different enemy behaviors.

![Figure 15.6 Cyberspace enemy concept](image)
- **Enemy Units**: enemy units require several, separate body parts. The general idea is for the enemies to look like a living tree with a head as the root and several branches in the form of different body parts.

  - **Head**: the enemy head should look generally circular with a large, glowing circle in the center. Like most things in cyberspace, the head should be mostly black but with a glow effect. This circle should change colors from the normal blue to an orange or red when the ability stored in the head of the enemy is damaged. The head should guide the branches/the rest of the body parts. The head should be connected to the body parts by a thin line. The line probably doesn’t even need to be too visible, since it should be obvious that the branches/body parts are following the head.

  - **Body Parts**: the body parts should look very similar to the head, but with slightly smaller circles. The body parts also shouldn’t be a single circle, instead it should look more like a group of circles vaguely ordered like a tree: with branches that lead to more branches and maybe leaves. In general, an enemy should have as many circles (something that doesn’t quite correspond to body parts or heads) as the number of behaviours they have. For the MVP, there should be several body part sprites: one circle, two circles and three circles. For the one and two circle body parts, it doesn’t particularly matter what orientation they’re in. The three circle body part should be oriented so that there is a single circle that’s always closest to the head. It’s also possible to make the 3-circle body part by combining or overlapping the one and two-circle body parts. Just like the head, the body parts should turn from blue to orange when they’re severely damaged.

**Nice-to-Have**

The Nice-to-Have version of Firewall includes both a few more tower types and different sprites for different types of enemies. This version also has some extra effects in the attack animations.

**Hero Units**

For the Nice-to-Haves, the hero will be able to place several different tower types in cyberspace: a short-range tower that uses a pulse to attack near by enemies, a long range tower that focuses a disruption beam on one enemy in its range and several yet-to-be-determined tower types.

- **Pulse Tower**: this tower needs a static image which will be placed on a cyberspace node. This image should be slightly smaller than the node so that the node can still be seen under the tower once the tower is placed on it. This tower should not be animated, but should have an animation of the pulse attack. This attack animation should look like a blue circle that starts out at full (100%) opacity and fades as it travels away from the tower. This can either be done by using a static circle that fades into completely transparent (0% opacity) and is scaled up using code, or a short animation by an artist. This pulse should have an additional effect such as particles that spread mostly at the same speed as the pulse, or an effect that looks something like an enemy being shocked when the pulse hits the enemy. As a Nice-to-Have, this tower should have several images of it with different levels of damage done to it. For instance, one for it at full health, another when it’s at 50% health and a final one when it’s down to 20%.
Finally, there should be an animation for when a tower is destroyed, though this can be the same animation for all towers being destroyed.

- **Long Range Tower:** this tower needs a static image that makes it easily distinguishable—beyond a simple color palette swap—from the other types of towers. Adding a long gun to the tower base or an equally visible weapon would be ideal. The base and weapon part of the tower should not be animated. Like the other tower, it should have an attack animation. This tower’s animation should be a line with a glow effect and particles that can be scaled so it reaches an enemy the tower is currently attacking. This line should also be distinguishable from the paths connecting nodes. This differentiation can either be from a different shade of blue or an extra glow effect on the line, but there shouldn’t be any player who confuses a path and a disruption beam. As a Nice-to-Have, this tower should have several images of it with different levels of damage done to it. For instance, one for it at full health, another when it’s at 50% health and a final one when it’s down to 20%. Like the previous tower, there should be an animation for when a tower is destroyed.

- **Nice-to-have Tower(s):** these towers, which we have yet to decide on, should all be easily distinguishable from the other two MVP tower types, again, besides being color-swapped from each other. Giving them the same bases with different weapons would be acceptable, or making them different shapes, depending on what makes sense with that particular tower’s abilities. These towers should have their own unique attack animations that make sense for their abilities. Like the previous tower, there should be an animation for when a tower is destroyed.

**Enemy Units**

For the Nice-to-Haves, enemy units will have a few animated images with slight variations in the sprites for different enemy types.

- **Enemy Units:** for the Nice-to-Have enemies, we will add icons to overlay onto the circles in both the head and body parts. These icons should, ideally, inform the player exactly what behavior is stored in that enemy’s node and allow them to try and select behaviors to destroy easily.
  - **Head:** the enemy head should look generally circular with a glowing circle in the center. Like most things in cyberspace, the head should be mostly black but with a glowing center reminiscent of the nodes in cyberspace. This circle should change colors from the normal blue to an orange or red when the ability stored in the head of the enemy is damaged. The head should guide the branches/the rest of the body parts. The head should be connected to the body parts by a thin line. The line probably doesn’t even need to be too visible, since it should be obvious that the branches/body parts are following the head. Most of the assets to show enemy type variation will be in the head. For instance, an AI with many attack behaviors should look more streamlined, or sharp, while an AI that might have defense or support behaviors could be larger or look more like a shield. The head should have an animation for when it is defeated.
  - **Body Parts:** the body parts should look very similar to the head, but with slightly smaller circles. The body parts also shouldn’t be a single circle, instead it should look more like a group of circles vaguely ordered like a tree: with branches that lead to more branches and maybe leaves. In
general, an enemy should have as many circles (something that doesn’t quite correspond to body parts or heads) as the number of behaviours they have. As a Nice-to-Have, we might have only one circle that can be procedurally drawn into the program, overlapping with however many other behaviors there are in that branch. Just like the head, the body parts should turn from blue to orange when they’re severely damaged. Each circle/behavior should have a death animation that doesn’t much affect the surrounding behaviors.

15.7 Hero Asset Needs

The hero unit will need two separate sprites, one for the physical world and another for cyberspace. Both of these sprites will have to be broken up into section to allow for easier animation.

The various areas that need to be separate parts that form the whole sprite are marked in the picture above. The red section is the main body of the sprite. This section will not any specific animation itself, but all other pieces will be attached to it and the weapons will rotate with it. The green section is the treads. This section will need a simple animation that shows the treads moving. The purple section is the minigun which is the hero’s main weapon. This section will need a firing animation that involves the minigun spinning. The blue section is the plasma cannon, one of the hero’s special weapons. This weapon will need a simple firing animation where the gun has a big kickback upon firing. The orange section is the rocket pack, another one of the hero’s special weapons. This section will also need a firing animation that has the rocket pack open to allow rockets to fire.

On top of these sections and their animations, the hero sprite will need a short animation that moves the sprite into an uplink state. This is the state the sprite goes to when the hero projects himself into cyberspace. The short animation will play and then the hero will remain in that pose.

The cyberspace projection of the hero will have similar asset needs to the physical. The main section of the sprite will have to be its own piece that the other parts of the sprite are connected to. This main section will need a simple animation for movement. In addition to the main section,
the cyberspace sprite will need two or three attached sections for weapons and abilities. These sections will need animations for coming out of the main section and then a couple frames that represent an ability being used or an attack happening.

15.8 Music and Sound Assets

15.8.1 Gameplay and Menu Sound Effects

Audio feedback is important in any interactive process and Firewall is no different. To create a fun and immersive game world our MVP will require a diverse range of sound effects.

Here is a list of sounds effects that we would need for our MVP:

- A sound effect for a single shot from a projectile weapon (that can be played for each shot of a semi automatic weapon or sniper rifle).
- A sound effect for rapid fire of a projectile weapon (that can be played for fully automatic weapons).
- A sound effect for a single shot from an energy weapon (that can be played for each shot of a semi automatic weapon or sniper rifle).
- A sound effect for rapid fire of an energy weapon (that can be played for fully automatic weapons).
- A sound effect for when a mech (friendly or enemy) is currently taking damage from weapon fire. This sound can be used for when the players base is taking hits as well.
- An explosion sound effect for when units are destroyed or when the player's base is destroyed.
- A sound effect for damage in cyberspace.
- An explosion for when something is destroyed in cyberspace, be it an AI behavior pod, the AI itself, the hero, the base, or a tower.
- A sound effect for the hostile energy pulses in cyberspace.
- A sound effect for placing of towers and firewalls in cyberspace.

In addition to those listed above, our nice to have gameplay sound effects would include:

- A different weapon fire sound effect for each weapon type in the game.
- Damage sound effects for each of the different mech types in the physical space.
- Separate explosion sound effects for units and the player's base in the physical space.
- Damage sound effect for AI behavior pods in cyberspace.
- Explosion sound effect for AI behavior pods when they are destroyed.
- Damage sound effect for AI core in cyberspace.
- Explosion sound effect for AI as a whole when it is destroyed.
- Damage sound effect for Hero in cyberspace.
- A different sound effect for each cyberspace tower, including firewall (sounds for when it fires and when it is placed).

There aren't too many sound effects that we require for our menus but what we will need is listed below.

The sound effects that we would need for the game play menus in our MVP include:

- One audio feedback sound effect for highlighting an AI behavior or a mech unit.
- A different audio feedback sound effect for when a highlighted behavior or mech is selected.
- One sound effect for when the menu is brought up or closed.
In addition, the nice-to-have sound effects would include:
- A different sound effect for each AI behavior when highlighting them.
- A different sound effect for each mech type when highlighting them.
- One sound effect for when the menu is brought up and another for when it is closed.

15.8.2 Background Music

Music is a very important part of any game. It can make or break the experience, and with Firewall, the case is no different. Firewall doesn’t require too much background music, but what it does require must enhance the overall gameplay experience. The music will help create atmosphere for the game, and will aim to provide a layer of aesthetic polish. By immersing the player into the game world, the soundtrack will effectively add to the story and game world in a way that the gameplay or any textual representation of back-story to the game can’t.

Since the game is based around playing in two worlds (the physical world and cyberspace), the music will reflect those two styles. We want to emphasize the “organic” aspect of the physical world, and we feel the best way to achieve this is to use an orchestral style with some electric guitar. Conversely, in cyberspace, we will be using electronic and dubstep inspired instrumentation to achieve our desired aesthetic.

We are drawing inspiration from various sources. Some game and film soundtracks that are inspiring the musical direction for Firewall are: Tron: Legacy, Ghost in the Shell, Deus Ex: Human Revolution, Frozen Synapse, and Battlestar Galactica (for some orchestral elements).

Our MVP calls for:
- One song at the title screen that continues through all screens up until the game starts.
- One song for the tutorial/introductory mode.
- One song for main gameplay.
- One song for the game over screen (losing the level).
- One song for the end credits (winning the game).

In addition, the nice-to-have background music would call for:
- Additional gameplay songs.
- Song dualities: Every orchestral-style physical world song would have an equivalent song written in more of a techno/dubstep style for the cyberspace world. This way, when the player switches between the two spaces, the song is structurally the same but has a different feel based on the new instrumentation to reflect the change in world.
- Voice Acting for units.
16. Concept Art

16.1 Concept Art Overview

- The physical landscape is a war-torn natural environment with remnants of battles of the past.
- The cyberspace landscape is a series of nodes that are either electric blue or electric purple on top of a black background.
- The style for the player’s mechs is going to be a combination of exposed machinery and smooth armor plating.
- The hero mech will be a similar style to the standard mechs, but it will be visually distinguished by its cockpit.
- The cyberspace hero will be recognizable as human in shape, but not material.
- The MVP enemy units will use the same sprites as the player units, but a nice-to-have would be to give them separate sprites that look more insect-like.
- The cyberspace enemies will have an abstract look to them in that they will be a shape with pods coming out of them that represent the different behaviors.
16.2 Physical Landscape

Firewall takes place in a lush world that is partially hidden by the remnants of past battles. As can be seen in the concepts above, the environments are littered with pieces of destroyed mechs and rundown buildings and ruins. This will show the player both the effects of the AI uprising on the world and humanity’s desperation. (See Section 3.2)
16.5 Cyberspace Landscape

In cyberspace, the landscape is very dark with strange structures that can be seen in the distance. The nodes and pathways between them are bright electric colors (see Section 3.3)
16.6 Physical Units
(Chapter 5)
16.7 Hero Unit (Physical and Cyberspace)
(Chapter 4)
16.8 Nice-to-Have Enemy Units
(Section 5.4, 5.5)
16.9 Cyberspace Enemies
(Section 8.4)
Glossary

**Area of Effect (AOE):** An ability or attack that affects an area rather than a single unit or single location. An example would be an explosion.

**Artificial Intelligence (AI):** A very advance computer program that can mimic intelligence and make complicated decisions. In Firewall, AIs control *mechs* in the physical world and must be destroyed in cyberspace.

**Cooldown:** A period of time that must pass before something can be done again. For example, a hero ability (in this game and in a game like DotA) usually invokes a cooldown that lasts anywhere from 2-180 seconds.

**Cyberspace Node:** On the cyberspace playing field, a point at which several pathways intersect. Towers and firewalls are constructed at cyberspace nodes.

**Cyberspace Pathway:** On the cyberspace playing field, a path that the cyberspace hero and enemy AIs can move along. Cyberspace pathways run between cyberspace nodes.

**Line of sight:** When a player and an enemy can “see” each other, or when a line drawn straight between each unit is unobstructed by any objects, whether they be other units or objects on the map.

**Mech:** A science fiction term for a large vehicle or robot, including ones on treads and animal shapes.

**Minimap:** A view presented to the player that is a smaller representation of the entire game map.

**Minimum Viable Product (MVP):** The MVP is the version of the game that the team feels is the absolute minimal amount of features that the game needs in order to be playable. When a feature is described as MVP (as opposed to Nice-to-Have) it means that it is a feature found in the MVP version of the game.

**Nice-to-Have:** A Nice-to-Have is a feature that does not quite fit in to the MVP. These are features that the team does not feel are required to create a complete experience, but, if added, would bring the game to new levels and increase the overall fun and player engagement.

**Non-Player Character (NPC):** Any fictional character not directly controlled by the player. In FIREWALL, this includes Artificial Intelligence (AI) units, both enemy and friendly.