# Table of Contents

<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1. Overview &amp; Vision Statement</td>
</tr>
<tr>
<td></td>
<td>1.1 The Hook</td>
</tr>
<tr>
<td></td>
<td>1.2 Overview</td>
</tr>
<tr>
<td></td>
<td>1.3 Vision Statement</td>
</tr>
<tr>
<td>_</td>
<td>2. Audience, Platform, and Marketing</td>
</tr>
<tr>
<td></td>
<td>2.1 Target Audience</td>
</tr>
<tr>
<td></td>
<td>2.2 Platform</td>
</tr>
<tr>
<td></td>
<td>2.3 System Requirements</td>
</tr>
<tr>
<td></td>
<td>2.4 Top Performers</td>
</tr>
<tr>
<td></td>
<td>2.5 Feature Comparison</td>
</tr>
<tr>
<td>_</td>
<td>3. Gameplay</td>
</tr>
<tr>
<td></td>
<td>3.1 Overview</td>
</tr>
<tr>
<td></td>
<td>3.2 Gameplay Description</td>
</tr>
<tr>
<td></td>
<td>3.3 Controls</td>
</tr>
<tr>
<td></td>
<td>3.4 Rules</td>
</tr>
<tr>
<td>_</td>
<td>4. Storyboard</td>
</tr>
<tr>
<td>_</td>
<td>5. Game Characters</td>
</tr>
<tr>
<td></td>
<td>5.1 Character Design</td>
</tr>
<tr>
<td></td>
<td>5.2 Types</td>
</tr>
<tr>
<td></td>
<td>5.2.1 Player Characters</td>
</tr>
<tr>
<td></td>
<td>5.2.2 Non-Playable Characters</td>
</tr>
<tr>
<td></td>
<td>5.2.2.1 Guards</td>
</tr>
<tr>
<td></td>
<td>5.2.2.2 Creatures</td>
</tr>
<tr>
<td></td>
<td>5.2.3 Artificial Intelligence</td>
</tr>
<tr>
<td>_</td>
<td>6. Level Design</td>
</tr>
<tr>
<td></td>
<td>6.1 Level 1</td>
</tr>
<tr>
<td></td>
<td>6.2 Level 2</td>
</tr>
<tr>
<td>_</td>
<td>7. Story</td>
</tr>
<tr>
<td>_</td>
<td>8. Media List</td>
</tr>
<tr>
<td></td>
<td>8.1 Concept Artwork</td>
</tr>
<tr>
<td></td>
<td>8.2 Sounds Summary</td>
</tr>
<tr>
<td>_</td>
<td>9. Technical Specifications</td>
</tr>
<tr>
<td></td>
<td>9.1 Sonar Waves</td>
</tr>
<tr>
<td></td>
<td>9.2 Fourier Transform</td>
</tr>
<tr>
<td></td>
<td>9.3 Artificial Intelligence Algorithms</td>
</tr>
<tr>
<td>_</td>
<td>10. Credits</td>
</tr>
<tr>
<td></td>
<td>10.1 Game Designers</td>
</tr>
<tr>
<td></td>
<td>10.2 Asset Producers</td>
</tr>
</tbody>
</table>

2
1. Overview and Vision Statement

1.1 The Hook

You are a secret agent sent into a hostile compound. However something goes wrong and you are captured, and you are tortured for information, but you don’t give in. As a last resort the terrorist chemically dissolve your eyes and leave you there, on the table, blind.

An hour passes and your torturers have yet to return. You manage to break free and stumble into the hallway in total darkness. You hear a noise to the left so you go to the right. The sounds of your footsteps illuminate the lower edges of the hallway, and a sharp sting in your head causes you to let out a groan that illuminates the entire hallway for just a second.

Scared, you enter the room next to you and the sound of the door closing behind you illuminates a dead body not two feet in front of you and the sound of feasting outlines a creature in the distance. Realizing and understanding that sound is the key to your sight.

You try to stay quiet and whisper your way to a hiding spot, stepping over bits and pieces of your torturer’s colleagues. Just as you think you’re home free, the morphine used on you wears off and you let out a deafening scream that reveals a scene so horrific, that those with the gift of sight would wish they were blind.

1.2 Overview

Sonar is a top-down 2D survivor game where you play as a blind secret agent that utilizes sonar (echolocation) to accomplish all of his tasks.

You will command your sonar abilities with the sounds that various other objects in the game (radios, footsteps etc) can make and of course, your own voice. With your arsenal of sound, you will be able to mislead creatures by throwing your voice with walky-talkies, intimidate the guards with ferocious creature calls, and command creatures with the sound of your voice to assassinate your targets.

When all your cards are down in total darkness, and its only you, your voice and a compound in which every being wants you dead, just how stealthy can you be while still making noise?

1.3 Vision Statement

How stealthy can you be while still making noise?
  • Use your voice and the sounds around you to navigate through a hostile compound.
2. Audience, Platform, and Marketing

This section will highlight the expected audience our game will be designed for as well as the platform it will be designed upon. Lastly, sales figures for games on the same market and expected revenue will be reviewed.

2.1 Target Audience

Our game is targeted towards people age 18-35. As for player gender, it will be ambiguous and will support both male and female play. The targeted geographic location will be those in the western world, such as Europe and North America. This is because of the use of terrorists as an antagonist, supporting what many believe to be a major threat to their survival.

2.2 Platform

Our target platform is the Personal Computer (PC) and Xbox Live Arcade (XBLA). These platforms were chosen because every one of the developers is familiar with Microsoft XNA and C#, making it easier to focus on making a fun game and less on the technological difficulties presented while working with other languages such as creating a content pipeline.

2.3 System Requirements

**PC**
- Decent graphics card that supports:
  - Shader Model 2.0 or higher
  - DirectX 9.0 or higher
- **Hardware:** Microphone Input
- **Helpful:** Xbox 360 Controller

**Xbox Live Arcade**
- Microsoft Xbox 360 Gaming Platform
- Xbox 360 Microphone
- Xbox 360 Controller

2.4 Top Performers

The following are game that have been successfully released and are similar to our game concept. These include Sonar, I MAED A GAM3 W1TH Z0MB1ES 1N IT!!!1, and FortressCraft Chapter 1.
Sonar
Release Date: October 31, 2011
Developer: Nerdy Bird
Publisher: Self
Platform: Windows
Sales: Unknown
Price: $1.99 USD
Description:
The game is a top down adventure game where you play a woman who is in a cave system with no lights and only a sonar scanner to see the world around her.
Images:

I MAED A GAM3 W1TH Z0MB1ES 1N IT!!!1
Release Date: August 16, 2009
Developer: Ska Studios
Publisher: Microsoft Game Studios
Platform: Xbox Live Indie Games (Xbox 360)
Sales: Over 200,000 copies by August 2010
Price: $1 USD
Description:
The game is a top down shooter that places up to four players against hordes of enemies stylized from other games. Enemies include zombies, asteroids, snakes and geometric figures.
Images:
FortressCraft Chapter 1
Release Date: April 8, 2011
Developer: ProjectorGames
Publisher: Microsoft Game Studios
Platform: Xbox Live Indie Games (Xbox 360)
Sales: Over 600,000 copies
Price: $3 USD
Description:
   The game is a three-dimensional sandbox game designed for players to build and create things as well as explore a world. It is very similar to Mojang’s Minecraft.

Images:

2.5 Feature Comparison

The closest top selling game to ours is known as Sonar and is listed above. The difference between our game and that Sonar is that the user is required to make noise through a microphone to produce the sonar waves that enable sight. Another difference is that our game is based on stealth and avoiding guards and other enemies, whereas Sonar is based around exploration and adventure.

Our game will sell over this because it takes the player to a whole new level of immersion with the microphone input allowing them to feel like they are the player character creating an engaging experience that hasn’t been created before on many levels.
3. Gameplay

3.1 Overview

Sonar focuses on making the player sneak around and escape the compound they are trapped without getting caught by the enemy. Visibility is limited to the player, only allowing sight through the player’s hearing. The game will be displayed with a dark screen, and will light up when any sound hits the player. Included in this is the player’s voice, which can be used by speaking into the microphone in order to illuminate the world in their vicinity. The player must find a way out without being caught, using noise and voice to navigate the halls while avoiding the enemy.

3.2 Gameplay Description

The game functions through the use of an Xbox 360 controller and a headset microphone. The player moves with the directional buttons of the Xbox 360 controller and interacts with objects in the world with the A button. In order to see the world around them, the player must use their voice to speak into the microphone. By speaking into the microphone the player will activate their sonar ability, shooting particles out that illuminate any objects that they hit. The player must find the exit of each level without being spotted and detained by the enemy. Each level has different objects and obstacles the player will have to deal with in order to succeed. Objects will be colored differently depending on their role to the player. Red will represent anything that is dangerous to the player. Green objects will be anything the player can interact with. The objects the player will have to use or avoid are:

**Enemy Guards**

Enemy guards will either statically guard a location or patrol an area. Upon spotting the player, an alert will go off telling all other guards that there is an intruder in the compound. The player must run and hide before the guards can get to them. If an enemy guard reaches the player then the player loses. Guards will react to any sounds that they can hear. If the player can reach the enemy without them hearing the player’s presence, then the player can perform a stealth kill. The enemy guards will be colored in red.

**Vents**

The player can enter ventilation shafts with the A button. Vents are used to move across the level without being seen. Vents are narrow corridors, limiting the view of the player and the amount of room the player can move in. Vents will be strategically placed throughout levels, allowing players to use them to either escape capture by hiding in them or avoid enemies by crawling through them. The vents will be colored green.
Radio
Radios will be used to allow players the ability to see different parts of the map without them using their own voice. Radios also mask the movement of the player, allowing the players to make more noise without the risk of being caught by nearby enemies. The player will have the option to turn on and off any radio they can reach. The radio will be colored green.

Hiding Spots
The player will be able to hide under or by objects in order to avoid being chased by the enemy. These spots will be colored green, and will represent different hiding areas like lockers, tables, and desks. The player can interact with these spots with the A button.

Creatures
There are other enemy NPCs present in the compound that interact differently with the player. Rather than try to capture the player, these enemies will try to kill them. These creatures will have different abilities and mechanics that the player must learn to defeat. All of these creatures are invisible to both the enemy guards and to the player; however they can be seen using the player’s sonar ability. They will also be able phase through the environment, such as through doors and walls. Instead of causing instant game over though, they will latch on to the player and possess them, causing constant damage. In order to break free of their hold, the player must use the microphone to make noises and scare them off. Each creature will have different abilities when they are attached to the player. See 5.2.2.2.1 for types of creatures.

Advanced Voice Commands
As the game progresses, the microphone will allow for many other uses other than the sonar ability. The player will gain different mechanics that will push the player to use their voice in interesting ways. These abilities will be unlocked through the game, and will have limited uses at a time. Some mechanics that are planned for additional microphone abilities are:

- **Growl:**
  By growling in the microphone, the player can control the creatures in the world. With this ability, players can change the target of creatures that are running after them, or summon a creature that they can use against the enemy.

- **Throwing Voice:**
  If the player uses different pitches for their voice, they can throw their voice in order to distract or lead the enemy. This includes causing sound to appear ahead of the player to get the attention of the enemy.

- **Yell:**
  When the player yells into the microphone, it could create a soundwave that will stun any guards or creatures around them temporarily.
3.3 Controls

The current control system supports an Xbox 360 controller and the headset microphone. The player can also choose to use any microphone that is connected to their console and keyboard controls. The controls for the Xbox 360 controller are:

- The directional stick is used to move the player around the level.
- The A button is used to interact with objects in the level. This includes vents the player can enter and hiding spots.
- The B button is used to subdue enemies the player can reach before he is spotted.
- The headset will be used for many different parts of Sonar. The player can use the microphone for the Sonar ability, allowing the player to illuminate the world around them. They can also use the microphone for many other additional features highlighted under the advanced voice commands section.

3.4 Rules

The player’s objective is to find the exit to each level before the guards find them. The level will have one start area and one exit. Each level will also have a different layout, consisting of different sets of hallways, rooms, enemy guards, objects, and creatures.

The player can only see from sound particles that hit them. Sound travels outward from the source, moving at a specific rate dependent on the source of the noise. Particles that hit objects will get reduced in loudness, and when the sound has been reduced enough it can no longer be heard.

Enemy guards will only patrol in their area or stay static while the player’s presence is unknown. When the player has been spotted, the enemies will run towards the player and try to find them until they get out of sight and hide. When the enemy guards reach the player, the guards will take them captive again, ending the level. The guards will also react to sounds around them based on the distance and intensity of the sound. If any sound particles from the player reach the guards with the right intensity, the guard will investigate the source of the sound. If the player can reach the enemy without the guard noticing them, the player can subdue the guard.
The player is only trying to reach the end of each level by any means necessary. There will be no sense of scoring mechanics, as the player can move at their own pace to simply find the exit. Once the player has made it through every level they will have won the game.
4. Storyboard

Demonstrates the core mechanics and basic visual representation of gameplay. The following begins with an introduction and then proceeds into the level.

Door is closed illuminating the world. The objects in the world fade away and leave you in darkness...

A notification appears and directs you to your microphone. After speaking into the microphone, the objects around you come into view.
Once the sound waves from your voice reach the green object in the corner, another notification appears letting you know that you can interact with green objects.

You interact with this object and duck into the vent. Once inside you have two options.
First you decide to go South. A guard notices you and you are sent back to first room.

You duck into the vent again and go East. You enter the break room with a radio playing and two enemies. A notification appears.

You turn off the radio. This alerts the guards and you are sent back to the first room.
After re-entering the break room, you leave the radio on and sneak past the guards into a hallway to the south.

You go through the door to the East and notice a sleeping guard. Instead of quietly leaving, you make noise into your microphone and this wakes the guard. This sends you back to the first room.
Instead of waking the enemy, you leave the room and continue down the hallway to the south. Here you notice three pathways and a paroling guard.

First, you avoid the guard and go down the East path. Here you run into a guard and have no way out. You are sent back to the first room.

You walk through again and decide to take the southern path. You end in a room full of enemies and a radio. You decide to leave the radio on to mask your footsteps.
You continue south. Here you notice an enemy to the East and continue to the West.

You check out the object at the end by speaking into the microphone. The stairway illuminates and you press A to interact with it. You have completed the level.
Now let's go back and see what happens when we quietly wait for the guard to pass.

You quietly walk past the enemy and take the Western path.

You run around the corner and down the hallway.
You see the stairway, representing the exit of the level and make noise to illuminate the stairway. You press A to complete the level.
5. Game Characters

5.1 Character Design

The Player Character
The player character is a human, who has had his eyes removed. He now sees the world through sonar, and the player must make noise in order to see the world around them. The main goal of the player character is to escape from the terrorist compound.

The Guards
The terrorists are the main non-playable character type. They will patrol the environment and investigate any strange sounds or events. They will also hunt the player character down if they see or hear him.

The Creatures
The creatures are a nice to have non-playable character. They will wander the environment, passing through walls and are hyper-sensitive to sound. Some creatures will run from sounds, others will run towards them. For more information see 5.2.2.2.

5.2 Types
Information on the various types of characters will be presented in the sections below.

5.2.1 Player Character
This section will cover the minimum value product as well as the nice to have features for the player character.

Minimum Value Product
- The player character can move throughout the world.
- The player character must be able to die
- The player must make noise to view the world.
- The player can move at various speeds in the world. This speed determines the footstep noise that projects sound, further illuminating the world.

Nice To Have
- The player will have health that will be decremented when hit and possessed. (See guards and creatures)
- The player can pick up and throw objects, which create noise in another location.
- The player can interact with the world. This includes, but is not limited to the opening and closing of doors as well as turning on and off radios.
- The player character will be able to be possessed by creatures (See 5.2.2.2).
- In order to free themselves from the possession of a creature, the player character will have to produce a constant loud noise.
5.2.2 Non-Playable Characters
This section will cover the minimum value product as well as the nice to have features for the various types of non-playable characters.

5.2.2.1 Guards

**Minimum Value Product**
- The guard chases the player throughout the world if seen. When chasing the player, the avatar will turn red, indicating that they are in chase mode.
- If a noise is too loud, the guard will investigate by patrolling that area.
- Must be able to beat or use a tazer on the player, defeating them and sending them back to the start of the level.
- The guards’ footsteps will make noise, illuminating the world around them.
- The guards will whistle, illuminating the world around them.
- The guards will also make a noise to signify they are on alert.
- Guards must be able to patrol certain areas.

**Nice to Have**
- Guards will interact with the world; turning on radios and opening doors.
- Guards will talk and start up conversations with one-another.
- Guards must be able to randomly wander through the world.
- Guards will use projectile weapons such as guns to attack the player from a distance.
- If an alarm is sound, guards will flock together from all directions and hunt you down.
- Guards will be able to be possessed by the creatures. (See Section 5.2.2.2)

5.2.2.2 Creatures

**Minimum Value Product**
- Creatures as a whole are nice-to have and are not essential to the game play experience.

**Nice to Have**
- The creatures will be invisible to the naked eye, and are only visible to you through the use of sonar.
- Creatures are hypersensitive to sound and depending on the type, will run away from the sound, or run towards it. (See types below)
- Unlike the guards, the creatures are able to pass through walls.
- Creatures will be able to posses both guards and the player.
5.2.2.2.1 Types of Creatures

Berserker
• Upon possession, the berserker creature will slam the character into walls.
• The berserker creature will run towards loud noises similar to the guards.

Tranquil
• Upon possession, the tranquil creature will slow you down, limiting your movement speed.
• The tranquil creature will run away from the source of loud noises, but will move towards the player character if it is in sight.

Silence
• Upon possession, the silence creature will deafen the avatar. This will make the player character not be able to see the world around them.
• The silence creature will be un-phased by loud noises until possession occurs.

Stone
• Upon possession, the stone creature will stop the player character movement completely.
• This stone creature will run towards the source of loud noises.

5.2.3 Artificial Intelligence
We will be implementing path finding algorithms using a variant of the A* algorithm.

Current Plan – We will be implementing a path finding algorithm using Hierarchical A*, which is a variant of A* that allows for changing of paths in the middle of graph traversal saving memory in the event of a needed path change.
6. Level Design

The following are a couple of levels that incorporate the NPCs that are included in the minimum viable product. This includes radios, guards (patrolling and wandering), a beginning, and an exit. The following is the key for levels:

- Everything that isn’t black is part of the level.
- White areas are parts of the level that aren’t initially visible.
- Yellow areas are areas being illuminated by sound. Darker yellow square is the source.
- Green block is the starting position.
- Red blocks are guards on patrol and are set at their initial position. Light red arrows indicate their patrol patterns.
- Blue blocks are wandering guards. Their path is indicated by the squiggly blue lines.
- Gray represents walls that sound can go through.
- Brown indicates doors.
- Lighter gray represents the exit.

(View levels on following two pages)
6.1 Level 1

This level introduces the player to patrolling guards. With the ample sound through radios or open windows, the player can freely run around without being detected. As they proceed further into the level, they must change their plan of attack and move slowly and quietly through the level, but still make sounds into the mic to figure out where things are located.
6.2 Level 2

This level requires the player to use the skills they learned in the first level to stealthily move through and to the exit. This level adds wandering guards, which have a random path to make it difficult to memorize their path and a vent way to sneak through the compound.
7. Story

This is a narrative explaining what is going on in the world and the story behind the player's existence. This narrative is still in development and will be modified in future quarters:

January 21st, 2019

If you are reading this, then I am surely dead.

My name is Dr. Nikolai Leonovitch, former head engineer of genetics for the now disbanded USSR's science division. In the motherland, my area of expertise revolved around discovering ways to genetically enhance the performance of soldiers on the battlefield, as well as discovering a way to increase a human's tolerance to radiation in the event of a nuclear war with America.

Thirty years ago, after the fall of the Soviet Union, researchers such as I found ourselves in a world that no longer needed us. While my research was originally able to provide a comfortable existence for my family, times soon became extremely difficult for us.

We lost everything.

It was at that point that I was contacted by an anonymous benefactor who said that his organization would fund my entire department in order to continue my research on genetics. Suffice it to say, I was ecstatic at the mere possibility, but one would have to be an idiot to not be skeptical of such an outlandish offer. Imagine my surprise when a group of men arrived in a helicopter at my doorstep inviting me to accompany them to my new laboratory.

I was transported to a remote island, not unlike that which you would expect to be the secret lair of the villain in a spy movie. Once I laid eyes on the facilities, it was difficult to avoid the comparison. The equipment was incredible...state-of-the-art equipment the likes of which the world had never seen and was decades ahead of its time. At that moment, I would have volunteered to conduct my research, and I was eager to begin work immediately.

Until I learned what was to be asked of me.

He wanted me to splice the genetics of animals into humans in order to create soldiers with certain desired traits. I was informed that my benefactor wanted to create a series of super soldiers that was more lethal and effective than any fighting unit than was ever assembled.
I refused without hesitation and he knew there was no chance of convincing me to willingly reconsider. I became a scientist to save lives, not destroy them!

How foolish I was to think that he could act with any semblance of humanity. It was readily apparent that the extent of this man’s monstrosity knew no bounds...he kidnapped my family and demanded that I do what he wanted or he would kill them.

In theory, the process of enabling a human to develop the traits of an animal is simple. We applied accurate doses of radiation to deteriorate the genetic structure of the test subject and then used stem cells to reconstruct the anomalies with the desired organs as a model. Initial trials were anything but successful. The entire first wave of test subjects developed horrible mutations to the point that they could no longer be recognized as human. The quantum nature of the radiation made their cellular structure severely unstable. They gained the ability to phase through matter and, and by using this property, they learned that they were able to “possess” human hosts. Furthermore, they vibrated on a frequency that avoided the visible spectrum. For these reasons, they were codenamed “specters.”

Each day, the crimes I committed against humanity weighed heavily on my conscience. My captor deemed the failed experiments unworthy, and ordered that they be euthanized and destroyed.

Though they were severely disfigured, they were still human. Unable to live with the guilt of further destroying a human life, I convinced him that they could still be useful to him. He agreed and demanded that I begin work immediately on the next group of test subjects.

Once we perfected the splicing process, later trials were “successful,” in his eyes. We were able to engineer and soldier with the eyes of a hawk. His purpose was to be a sharpshooter, able to snipe from great distances without the use of a scope. We were successful in splicing a soldier with the genes of a crocodile who was cold blooded and undetectable by infrared optics. Our most recent experiment centered on creating a soldier who was to have perfect night vision. It was decided that he would be spliced with the genes of a bat in order to be able to maneuver by echolocation...

Given the atrocities that I’ve committed, I cannot consider myself fit to continue to draw breath.

There is no doubt in my mind that my family is already dead. I only wish that I could beg for their forgiveness, but if there is such a thing as Hell, then I am to remain there for the rest of eternity. I have committed to sabotaging my lab and eliminating any trace of the horrors that I have made possible.
8. MediaList

8.1 Concept Artwork

8.1.1 Player Character Concept Drawings

The following drawings are not precisely what we will be going with, but give a representation of what the art style will be.

Characters, weapons, and dead bodies
Player and Enemy Mockup
Creatures
Kraken Creature
8.2 Sound List

8.2.1 Environment

- Throwing a rock (initial throw)
- Rock hitting ground (possible bounce across)
- Ceiling Fan
- Floor Fan
- Air Conditioning / heater Motor
- Halogen lights (buzzing noise)
- Light switch flick
- Opening hinged wooden door
- Closing hinged wooden door
- Kicking wooden door open
- Opening hinged steel door
- Closing hinged steel door
- Attempting to kick steel door open
- Run into concrete wall
- Run into wooden wall
- Radio song (plays on all radios)
- Carpet floor (dark and muffled)
- Wooden floor (dark and echo-y)
- Metal floor (echo-y and bright)
- Marble floor (echo-y)
- Wet floor
- Stepping on twigs
- Stepping on dead body

8.2.2 Guards

- Talking / Conversation
- Whistling
- Heavy footsteps
- Guard comments:
  "Hey!"
  "Capture the prisoner!"
  "What was that?!
  "That's strange..."
  "What was that noise?"
  "I could swear I heard something..."
  "Hey you!"
  "Don't let the prisoner escape!"
- Gun shot (single shot)
- Gun shots (automatic)
• Quick kill (no noise from guard, collapsing to the ground)
• Quick Painful kill (short scream from guard, then collapse to ground)
• Long painful kill (long scream from guard, then collapse to ground)
• Dragging sound (dragging dead guard around)
• Dropping sound (dropping dead guard a foot to the ground)

8.2.3 Creatures

• Screeching while dying (when player is screaming them out of themselves)
• Possession of player
• Possession of guard
• Death noise (disappear)
• Footsteps
• (more to come)

8.2.4 Player

• Footsteps (walking)
• Footsteps (running)
• Scuffling (crawling)
• Heavy breathing
• Heart beat (slow)
• Heart beat (fast)
• Take out knife
• Knife guard
9. Technical Specifications

9.1 Sonar Waves

The vision in Sonar is portrayed to the screen in ripples emanating from each sound source, granting the player vision of objects that they pass over. As such, in order to gain a view of their surroundings, the player must create noise, either through microphone input, or through other in-game methods, in order to view their surroundings. When implemented alongside enemy behaviors, this will create many interesting areas where the player is forced to balance their actions between detecting their enemies and avoiding them. In addition, it should work to create an interesting visual effect for the player.

Some current challenges in implementing this effect are how to deal with objects that are large enough only to be partially lit, and how deal with the processing problem of having to calculate how visible each object should be each frame, given how far away they are from the sound source. We have come up with a fairly sufficient solution to the second problem, and are hopeful in finding a solution to the first.

Below is the current algorithm as we are implementing it in our prototype, described in pseudocode.

Set up:

        global int soundwaves[]
        global list <VisibleObject> declaredobjects //list of all declared objects.

Update:

     for each list in ObjectDistances:
          list = empty list

     for each VisibleObject in declaredobjects:
          int D = roundDown(distance between player and VisibleObject)
          //use this list for other purposes as well, such as collision detection
          ObjectDistances[D].append(VisibleObject)

     for int i = MAX_SOUNDWAVE_DISTANCE; i>0; i--:
          soundwaves[i] = soundwaves[i-1]*sound_decay_rate
          soundwaves[0] = current_sound_strength

     for int i = 0, i<MAX_SOUNDWAVE_DISTANCE, i++:
          for each object in ObjectDistances[i]:
              object.alpha = (soundwaves[i]-reduction(i))/(wallRaytracer()+1)
9.2 Sound Analysis

9.2.1 Decibel Analysis

The first sound analysis feature calculates the decibel value from sound taken from the microphone, allowing the user to give specific commands based on the volume of their voice. XNA 4.0 can easily provide decibel recognition by implementing the Microphone class and the Audio namespace (Microsoft.Xna.Framework.Audio).

This mechanic will be implemented in two ways. The first is merely to produce sight around the player’s avatar. The user is able to see the game world through the use of sound given through the microphone. The area around the player’s avatar will illuminate when sound is given, and the decibel value determines the radius and brightness of the illumination. The advanced mechanics of sight, as well as pseudo code, is explained more in-depth in a different section of this design document.

The second implementation of decibel analysis is under the pseudonym “growl mechanic.” The growl mechanic analyses decibel values given and checks if they are within a specified range. One specific growl mechanic is implemented to test for very high decibel values. When a user provides very loud input into the microphone, a large growl sound file is played through the speakers over the player’s voice. This sound then provides the player with a very large range of sight, as well as alerting all other characters within a certain range. A large characteristic of the growl mechanic is playing a similar or desired sound over the player’s voice. This gives the appearance that the player is creating the sound themselves.

Within these implementations the decibel values from the microphone are stored within a buffer. This allows for the buffer to be used for both decibel and frequency analysis, without duplicating data. Frequency analysis, and how it uses this, is explained in the Frequency Analysis section.

![Audio Input](image)

- High Decibel
- Low Decibel
- Time
9.2.2 Frequency Analysis

The second sound analysis feature calculates and analyzes the frequency from given sound input. By calculating the frequency, the input can be categorized into certain pitch levels, allowing for more complex actions and commands. To obtain a pitch, a frequency spectrum (a graphical representation of frequency in respect to time) must be generated using fast Fourier transforms of the audio. Since there is no function or class within XNA 4.0 that calculate frequency or pitch from audio, an outside library will be used to accomplish this task. The library will do the following computations:

The initial audio will be stored based upon its decibel value. The computations and code associated with this step should already be explained in the previous section titled decibel value. Sound Analysis Figure 1 visually represents the audio input received from the microphone. These values are stored within a buffer of a preset length (must be a power of 2). The size of the buffer determines the accuracy in determining the frequency of the sound while also having a longer execution time.

The samples within the buffer are then sorted in bit-reversed order and placed into a new buffer of the same size. To sort in bit-reversed order, integers within the buffer are converted into binary. They are then split into two groups depending on whether the Binary number ends in one or zero (those with zeros precede those with ones). These two groups are then each split, using the same condition, except with the second to last digit. This continues until the final groups are sorted based upon their first binary digit.

These sorted digits are placed into a new complex number buffer of the same length. When creating a frequency spectrum, the frequency value (y-axis) uses complex numbers as units. Complex numbers consist of a real part and an imaginary part, and can be visually represented as a pair of numbers forming a vector on an Argand diagram. Sound Analysis Figure 2 is an example of the aforementioned image. The formula for a complex number is $z = x + iy$, with $z$ being the complex number, $x$ being the real part, and $iy$ being the imaginary portion. A complex class already exists in the Microsoft Windows
operating system and will either be used or duplicated and slightly modified for this buffer.

Once the initial input is resorted and placed into a complex number buffer, a digital fast Fourier transform can occur on the buffer. For clarification, the length of the buffer (which should be a power of 2) is the number of points the transform will do. The fast Fourier transform will treat the buffer as a vector of a number of dimensions equal to the buffer length. The average Fourier transform process will work in order of the buffer length squared \((O(N^2))\). However, for a more efficient and quicker transformation, fast Fourier transforms will be used within this frequency analysis, which work in the order of the buffer length multiplied by its own logarithm \((O(N \log(N)))\). While both types of transform use the same formula, fast Fourier transforms use a divide-and-conquer method to speed up the process. The following equation will be used for the transformation, with slight alterations if needed:

\[
V[k] = \sum_{n=0..N-1} W_N^{kn} v[n] \text{ with } W_N^{kn} = e^{-2\pi i*n/N}
\]

This is the equation that will be performed on all complex numbers within the buffer. The actual implementation of the equation can be represented graphically using a butterfly calculation. Figure 3 is an example of a two point fast Fourier transform represented graphically:

![Sound Analysis Figure 3](image)

The complex number buffer in Figure 3 is named \(v\) with the output being stored in a buffer named \(V\). The example shows the equation being used twice \((V[k] = \sum_{n=0..2^-1} W_2^{kn} v[n])\).

With a buffer containing more than two elements, a fast Fourier transform will execute the equation in multiple rounds. A four point transformation, for example, consists for two rounds of execution. The number of rounds is determined by the length of buffer in terms of its power of two. The value of its power is the number of rounds the equation is performed. Since a four point transformation consists of two levels of butterfly calculation, its diagram is represented in two parts. Figure 4 is an example of a four point fast Fourier transform represented graphically.

Finally, to further show the complexity of the fast Fourier transformation applied to a larger sized array, Figure 5 is an example of a 16 point fast Fourier transform. In this diagram, the ordering is flipped, with the input of the diagram in sequential order and not the output. The components of the figure, though, are the same as the others. It is worthy to note that to determine which two complex numbers are transformed together
during the first round is determined by the buffer length. The index value of the two buffers differs by exactly the length of the buffer divided by two. So, with a buffer length of 16, the first two complex numbers are found at indexes 0 and 8.
At this point, while the fast Fourier transforms have been executed completely, the initial goal of creating a frequency spectrum has not. The output of the fast Fourier transforms are the amplitudes of a given sound at varying frequencies. Instead, these values must be displayed in terms of time. Figures 6 and 7 display the direct output of the fast Fourier transforms with figure 7 having the addition of time.

![Fast Fourier Transform Output](image)

**Sound Analysis Figure 6**

Figure 6 displays the different frequency amplitudes clearly at a given moment. The decibel values that were initially used for the Fourier transforms directly influence the amplitude values. A quieter sound will have overall lower amplitude values (closer to the x-axis) while a louder sound will have higher amplitude values (further away from the x-axis). The different x values of the figure display different frequencies. Deeper and lower frequency sounds have amplitude spikes on the leftmost part of the graph. Sounds that are higher pitched have larger amplitudes toward the right portion of the figure.

The main difference between figure 6 and 7 is the addition of time. While figure 6 can accurately and clearly display differences in amplitude in reference to frequency, the figure would change every moment sound is analyzed. Therefore, unless the input is recorded, figure 6 is a completely dynamic graph that is constantly changing.

Figure 7, on the other hand, is a three-dimensional graph that displays frequency, time, and amplitude. To simplify the visual aspect of the graph, the third axis, amplitude, is removed and is instead, replaced with a color key. The lighter the color is, the lower the amplitude of that specific frequency is. If there is no color, then the amplitude value at that given frequency is zero. The darker the color is, the higher the amplitude value. The advantage of figure 7 is that the changes in frequency amplitude over time allows for more complex frequency analysis. Instead of checking for a specific pitch at one moment in time, changes in pitch over a certain given time can be checked and analyzed.
Frequency analysis allows for a variety of new commands that a user can enter that are unavailable with decibel analysis. The first implementations of this feature will check for a user given sound with a continuous pitch within a certain range over a specific time length. The desired pitch would be previously implemented and played within the game for the user. This allows the user to listen to the sound and attempt to mimic it, creating a specific effect. Failure to stay within the desired pitch range would result in the normal effect of making sound.

After simple pitch recognition is implemented, a more advanced feature will be created. This feature will check for changes in pitch over a certain time. Once again, this would be provided within the game to allow the player to listen to it and memorize the sound. The player could then attempt to mimic the sound and its fluctuations. This advanced feature allows the user to learn how to control the pitch of their voice, and adds a level of difficulty and complexity to different game play elements.

The most advanced pitch recognition mechanic is to analyze the changes in pitch, over a specific time frame, checking for specific words. The complexity of this feature deems too cumbersome to accomplish within the timeframe given while also implementing other game features. It was advised by numerous professors that, while it would be interesting, word recognition should be avoided and not strived for.
10. Credits

10.1 Game Designers

The following is a list of the game designers contributing to the development of the game. Next to each name is their specified job during production.

- Chris Peterson: Graphics Programmer
- Corey Dimiceli: Technical Director
- Derrick Huey: Artificial Intelligence Programmer
- Devon Wyland: Producer
- Evan Weintraub: Graphics Programmer
- Joshua Ray: Lead Game Designer
- Ryan Anderson: Artificial Intelligence Programmer
- Steven Ekejiuba: Co-Technical Director
- Travis Carlson: Game Designer

10.2 Asset Producers

The following is a list of the artists, composers, and writers who will be helping create assets for the game throughout the year. Next to each name is their specified job during production.

- Asma Uz-Zaman: Artist
- Fenyang Smith: Composer
- Harrison Okins: Artist
- Jacob Pernell: Composer
- James Richards III: Writer
- Masami Kiyono: Artist