MicroVentures

A procedurally generated story driven retro adventure game for mobile devices.

CMPS 170 - Fall 2011
MicroVentures Design Document

Description

MicroVentures is an adventure game for modern mobile devices. Each play-through is a new & unique procedurally-generated experience that lasts 2-4 minutes, the average duration of most casual mobile games. Advanced narrative AI connects each adventure to previous play-through, creating a riveting interactive storytelling experience.

Table of Contents

Description 2
MicroVentures Team 3
Target Audience 7
Platform 8
Game Influences 10
Basic Gameplay 13
Fictional World Layout 16
Procedural Level Generation 20
Procedural Story Generation 26
Art Style 38
Sound Style 44
User Interface 46
High Level UML Diagram 52
Prototypes 53
Storyboards 57
Letters of Commitment 75
Conclusion 76
Our team will be broken up into three main categories for the development of our game, which each member having both a primary and secondary position in the following positions:

**Graphics/UI Programmer:** These members are responsible for the UI as well as tech graphics. For our game, this entails drawing the textures on the phone screen, I/O capturing through the touchscreen, game states, menu options, and of course, the game user interface.

**Gameplay Programmers:** These members are responsible for the gameplay aspects of the game. For our game, this entails the combat system, game balance, items, enemy AI, etc.

**Procedural/Engine Programmers:** These members are responsible for building the game’s engine. For our game, this entails creating the procedural engine that will build the procedural story and game generating system. This will also include the group that researches Minstrel story generation for use in our game.

---

**James Farmer:** Project Lead

Excellent programmer in C++, C, and Java. Some experience with procedural story generating, including both Universe and Minstrel. Experience in OpenGL, Marmalade, Android SDK, Web Development, and much more.

*Primary Group:* Procedural Engine  
*Secondary Group:* Graphics/UI Programmer

---

**W. Fletcher Cole:** Lead Engineer

Much experience programming in C, C++ and Java. Has researched and experimented with procedural level generation and studied procedural storytelling. Has experience working with teams on games.

*Primary Group:* Procedural Engine  
*Secondary Group:* Gameplay Programmer

---

**Zakary Sheridan:** Producer

Experience with C++ and UML diagrams, studied procedural story generation

*Primary Group:* Procedural Engine  
*Secondary Group:* Graphics/UI Programmer
Kyle Huey: Programmer / Fiction Liaison

Strong in C, C++, and Java. Experience writing and planning for games with branching storylines.

*Primary Group:* Procedural Engine  
*Secondary Group:* Gameplay Programmer

Sam Jenkins: Programmer

Can program in C++, C, C#, Java, Python. Also knows XNA.

*Primary Group:* Gameplay Programmer  
*Secondary Group:* Graphics/UI Programmer

Mark Zablan: Programmer / Procedural Specialist

Excellent programmer in C++, C, and Java.

*Primary Group:* Procedural Engine  
*Secondary Group:* Graphics/UI Programmer

Dan Healy: Programmer / QA Specialist

Proficient in C#, C++, Java, C, Python, Unity, Inform 7, and Wide Ruled.

*Primary Group:* Gameplay Programmer  
*Secondary Group:* Graphics/UI Programmer
**Gerrit Eggink:** Programmer

Has experience programming in Java, C++, C# (XNA), and C. Previous projects include designing advance algorithms, programming game physics / collision detection / user interface, and working with low level designing.

*Primary Group:* Gameplay Programmer  
*Secondary Group:* Procedural Engine

---

**Clement Tran:** Programmer / QA Specialist

Excellent programmer in C++, C, and Java.

*Primary Group:* Gameplay Programmer  
*Secondary Group:* Graphics/UI Programmer

---

**Christopher D. Garcia:** Programmer / Art Liaison

I have experience programming mobile games with Android and am strong in Java, C++ and have worked with ActionScript, C, C#, HTML, PHP, JavaScript, SQL, Python, Perl, Scheme, oCaml, SmallTalk, Prolog; rapid prototyping with GameMaker / Flash / Microsoft XNA.

*Primary Group:* Graphics/UI Programmer, Art Manager  
*Secondary Group:* Procedural Engine
Artists

These are members who are responsible for the art aspects of the game. Each of these artists have signed a letter of commitment for MicroVentures, which can be found in the Appendices at the end of this document.

Elmer Zhu
Art & Graphics

Eric Lawson
Composer

Xian Chua
Art & Graphics

Lauren Scott
Writer

Kimberly Shannon
Art & Graphics
Target Audience

Our target audience is primarily mobile device users, since this is the platform we aim MicroVentures to be on. Going deeper, we aim for this game’s audience to be casual gamers, or rather users who maybe don’t have the time to play through a long, cumbersome video game. We aim for each play through to be 2-4 minutes since this is a rough time estimate of a line at the airport, a coffee break, etc. We aim for each adventure to be fully played through during each play session, and the 2-4 minute range will not be exceeded unless the player decides to play another adventure right after.

We understand that most people who play games on their smart phones are busy and probably don’t have time for a truly engaging story. They instead would rather kill a few free minutes with a simple game such as Angry Birds or Fruit Ninja. Our game is aimed at the audience of mobile device users who are bored by the mindless nature of most casual games. Our ultimate goal is to create an ongoing, engaging story that is given to the player in bite-sized pieces during each adventure.

We are also aiming to capture the original Roguelike audience. Even though the gameplay may not have as much depth as Rogue, NetHack, even current mobile Roguelikes such as Legend Of Yore, the amount of gameplay in each adventure in MicroVentures is practical when we balance both the nature of Roguelikes, a single playthrough of a one-time dungeon, with the realistic time span one has on a mobile phone, which could be less than 5 minutes when on the go. We will make up for this shortened gameplay with story arcs and references to previous adventures. We believe that by connecting these self-contained short adventures with story, the game has incredible replayability and a rich context that makes play meaningful.
This game is being targeted towards smartphone devices, and will first be built to run on the iPhone. Since we are using the Marmalade SDK (www.madewithmarmalade.com) we will be limited to developing for iPhones with iOS 3.0 and above. Below is a pie chart specifying the most recent disbursement of iOS versions for iPhone users. According to the demographic, the Marmalade limit of iOS 3.0 and above will not be a problem, as only 0.17% of iPhone users own iPhones that run under version 3.

Sources:
- http://www.madewithmarmalade.com/marmalade/supported-platforms
Marmalade

We will be using the Marmalade SDK to develop MicroVentures. Marmalade is a set of software tools that allows us to develop for multiple mobile platforms at once, including iOS, Android, and Blackberry. Instead of using a separate SDK for every mobile platform, we are able to use Microsoft Visual Studio and program in C++ as if we were developing for a PC game. We just need to include some of Marmalade’s library functions for display and marmalade automatically ports our code to whichever mobile device we want to develop for.

One of Marmalade’s most useful features is its built in simulator. The simulator simply runs from Visual Studio and lets us see what our work looks like on a mobile device screen without having to deploy a test program to the device every time we test a feature. The simulator also allows us to gather data such as memory usage and simulate device specific features like screen resolution, memory, and audio and video codecs.

For our group, we believe that Marmalade is the best choice for our SDK for several reasons:

1. It uses C++. As CS Game Design majors, we are all very familiar in this language, so although we will be building our game on a completely new platform for us, we will not have the additional burden of learning another computer language.

2. Marmalade has proven its worth! Syz E.G., Grand Prize winner of last year’s Sammy’s Awards, was completely built on Marmalade. We have personally been shown the developing environment and its capabilities from Team Krinoid, and we feel confident enough in its capabilities.

3. Marmalade is Cross Platform, meaning we will be able to develop for multiple mobile devices, giving us more room not only for game testing, but also to learn more about current mobile devices as a whole.
Game Influences
These are some of the games and genres that influenced MicroVentures

Roguelikes
A very popular genre among game enthusiasts, Roguelike (also called a Dungeon Crawler), is the gameplay genre of choice for MicroVentures. Wikipedia states that Roguelikes are “characterized by randomization for replayability, permanent death, and turn-based movement.” Famous Roguelikes include Rogue (the original), Nethack, even Diablo. MicroVentures is essentially a Roguelike, though the experience of MicroVentures is very shortened.

Desktop Dungeons
This game is a very simple Roguelike which demonstrates some of the basic characteristics which will also be present in some form in MicroVentures, including procedurally generated levels and pixel art. However, MicroVentures will generate smaller dungeons to be completed in 2 to 4 minutes, and will use item acquisition to improve the character’s abilities, rather than a level-up system.
Legends of Yore
This game is a popular Roguelike for mobile devices, and is going to be a major influence on the way we set up our game for a touchscreen phone. It uses movement with touching, and also has pathfinding for when you select a location to go and objects are in the way. We are aiming to implement movement this way. However, the two things we want different with our game is the screen following the protagonist, which leads to jerky screen movement, and the amount of options and information on the screen. We want for the player to be able to zoom in/out, drag around the camera, and we also want to keep the UI to a minimum.

The Binding Of Isaac
The Binding of Isaac game with procedurally generated levels where items are used to give the player abilities during play. We were influenced by The Binding Of Isaac because it utilizes Roguelike features while staying a simple, stat-minimum game. Like this game, the characters in our game will have no leveling, but rather will strengthen up by picking up a number of items available in the game, all changing the gameplay dramatically. The Binding of Isaac currently has 131 items, and for our MVP we are shooting for at least 40 items, but we’re positively going to have more than that.
Mobile ‘Casual’ Games
Various casual game for smart phones that use the touch screen as a controller. Casual games offer players short, non-committed gameplay that engages them for a short time. One of the things we feel is missing from these Casual games is story: it’s extremely hard to fit a good story within such a small structure. We plan on implementing story within our casual gameplay by having the story in between each play-through of an ‘Adventure’.
Basic Gameplay

Story Intro
The gameplay for this game is going to be simple and Roguelike. Before starting a session of MicroVentures, The player will be presented with a portion of story text. This story text will tell the player what playable character he or she is playing now (The Warrior, Wizard, or Rouge), where they are in the world, and what they need to accomplish. Depending on choices made in previous games, the story will also tell what has happen to the character since the last play and why they are trying to accomplish the goal. For more details on how the story affects the gameplay, read through the Story Generation section of the design document.

World and Movement
The game world and character movement will be similar to Desktop Dungeons. The game world will be constructed with a number of square tiles. The player, monsters, chests, and goal items will take up a tile each while the final boss will take up four tiles. Each tile will be either passable or impassable terrain. The tile’s design will be different depending on where the character is in the story. For more information about levels, read the Level Generator section of the design document. When the game starts, the player will be able to see the tile he or she is on and all tiles 2 tiles away from the character horizontally, vertically, and diagonally. All other tiles will not be displayed. The player can move onto any adjacent passable terrain horizontally or vertically per turn. Tiles that are occupied by a monster or final boss are considered impassable until they are dead. See the Statistic and Combat portion below for information about how to kill a monster. When the player moves, all tiles 2 tiles away from the character horizontally, vertically, and diagonally will be displayed.
**Statistic and Combat**
Each character, player/monster/final boss, will only have two statistic variables: ad and health. Ad, or attack damage, is an integer that represents the power a character has. Health is an integer that represents how much damage a character can take before dying. To engage in combat, a player moves onto the tile occupied by a monster or the final boss. The monster can also move onto the player. Whenever a player is in site of a monster, the monster will move towards the player the same way the player does. The monster can only move two squares away from its initial position. If the player leaves that area, the monster will return to its original position. Other creatures might run away when in site of the player. When in combat, a random number from one to the player’s ad is subtracted from the monster’s or final boss’ health and vise-versa. If the health variable for a character is 0 or lower, that character die. If neither character died, the player is moved to an adjacent visible passable terrain next to the monster.

**Items and Character Development**
The character develops similarly to The Binding of Isaac. In order to make the player character more powerful, they must find items that raise their stats. Items are either found in chest or dropped by monsters; Goal items are the only exception. Health potions will recover a portion of a player’s missing health. There will be weapons, armor, and tomes that increase a character’s base statistic variables. Miscellaneous items will affect the world or player in different ways (such as an invisibility cloak that will be used to sneak past an enemy, or a teleporter that will move the player to a different part of the level). There will also be a goal item the player must receive to finish the game, and one or more preset items that must be acquired before reaching the goal item. For a full list of each item and their use, read through the Item List of the design document.
The End Game
There are two ways to end a session: the player dies or a goal item is taken. Once the player’s health reaches 0, they fail the quest and end the game session. The story generator stores that playable character’s death in the story bank for future reference. That playable character will be unable to play for a while. When a goal item is received, the player wins and that playable character is stored in the story bank for future uses. The playable character’s items and statistic variables are updated and the character will be used in a future game. In both cases, another portion of the story text is shown to the player. The story tells the player what they did in the game, if the player died or found the goal item, and what the playable character will do next (unless they died). The whole story is then updated and the game returns to the menu screen. For more information about the story bank and its uses, read the Story Generator section of the design document.

Tutorial Level
There will also be a tutorial level in which the various game mechanics are introduced during a single adventure. This tutorial level will always be the first level in a freshly started Story Bank. We also would like to have possible different endings to the tutorial to start the Story Bank off in a specific direction.

For Storyboards on a typical adventure playthrough as well as a the tutorial, Go to the Storyboard section on Page 57
Fictional World Layout

Story Arcs
These are the overarching goal of the hero’s story and the final dungeon. May not always be referenced by adventures, but gradual buildup to the final adventure will occur.

<table>
<thead>
<tr>
<th>Arc</th>
<th>End Goal</th>
<th>Heroes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEEKER OF TRUTH</td>
<td>Find something that has intrinsic value to the hero (a family heirloom, a long lost mentor or family member, a cure, a place, etc)</td>
<td>Wizard</td>
</tr>
<tr>
<td>LEGENDARY</td>
<td>Perform an action that gains you legendary status</td>
<td>Warrior</td>
</tr>
<tr>
<td>TREASURE HUNTER</td>
<td>Attain great personal wealth and power</td>
<td>Rogue</td>
</tr>
<tr>
<td>VANQUISHER</td>
<td>Defeat the ultimate evil and bring peace to the land</td>
<td>Rogue, Warrior, Wizard</td>
</tr>
</tbody>
</table>

Templates
These are possible story templates we will be using for

<table>
<thead>
<tr>
<th>Template</th>
<th>Goal</th>
<th>Heroes</th>
<th>MVP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAPTURE</td>
<td>Pursue and catch an enemy or npc</td>
<td>Rogue, Warrior</td>
<td>N</td>
</tr>
<tr>
<td>COLLECT</td>
<td>Gather many items to activate something</td>
<td>Rogue, Warrior, Wizard</td>
<td>N</td>
</tr>
<tr>
<td>CONTEST</td>
<td>Compete with npc to kill most enemies, gather most items, or beat boss fastest</td>
<td>Warrior</td>
<td>N</td>
</tr>
<tr>
<td>COUNTDOWN</td>
<td>Defeat boss or destroy item before time limit</td>
<td>Rogue, Warrior, Wizard</td>
<td>N</td>
</tr>
<tr>
<td>DEFEAT</td>
<td>Defeat the boss</td>
<td>Rogue, Warrior, Wizard</td>
<td>Y</td>
</tr>
</tbody>
</table>
### The Heroes

For the MVP, there will be three main characters available.

#### The Warrior

"I’ll slay anything that’s evil. That's my deal."

A pillar of resolve and justice, The Warrior’s sole desire is to be a hero. Wherever there’s princesses who need saving, or kingdoms that need defending, he is always the first to be summoned by the King. However, his idealism is also his greatest weakness - The Warrior is young and naive. To him, evil is an unpredictable anomaly in the harmonious tapestry of good and righteousness in the world, not a constant threat that must be faced on a day-to-day basis.

#### The Wizard

"Mastery is achieved when telling time becomes telling time what to do."

Beneath a quiet, refined demeanour lies an unfathomable well of mystery. The Wizard is never done learning, and therefore is on a never ending quest for knowledge about the surrounding world and all it’s mysteries. The wisdom of his travels and the great power he wields make him a valuable ally - unfortunately, his abilities make him a pompous know-it-all. His tendency to speak melodramatically and explain everything in layman’s terms doesn’t make him popular either.
The Rogue
"I've always equated ‘feelings’ with ‘getting caught’...they both get in the way of my money.”
An aloof shadow, The Rogue has no loyalty to any kingdom or flag, only to whoever can pay his price. He is a greedy bastard, concerned only with material wealth and never higher callings or the suffering of the less fortunate. Second only to his greed is his wit - one does not survive long on the streets without a mind as sharp as his blade. Of course, he is frequently baffled by the idiosyncrasies of his world and its natural laws (he breaks the fourth wall a lot)

Nice-To-Have Heroes

The Princess
"What, you think a feudal warlord's daughter just sits around arranging flowers all day?"
The epitome of “An iron hand in a velvet glove”, The Princess plays the part of demure lady by day, and warrior by night. She secretly trains in the ways of war every night, and evens goes out on quests of her own. She is a bit of an oddball, but she has the love of her people and the fear of her enemies.

The Orc
"If you're going to kill something, you might as well kill it a whole lot.”
A blunt warrior inspired to greatness by his brutal defeat at the hands of the main heroes, The Orc makes up for his lack of grandeur with overwhelming force. He guards his tribe's lands from any and all threats, but is torn between loyalty to his tribe and friends, and his new found friends on the side of good.

Enemies
Small, medium, large, and boss enemies
- **Small**: mice, bugs, new enemies we'll think of - drop minor items
- **Medium**: unarmed or lightly armed humanoids, enemies that can inflict light curses/ailments/poisons, new enemies - drop regular items
- **Large**: armed warrior humanoids, giant lizard type, enemies that can inflict moderate to severe curses/ailments/poisons, new enemies - drop valuable items
- **Boss**: Huge slime, dragons, archmages, legendary warriors, new enemies - drop legendary items
NPCs
For NPCs that spawn within dungeons, I would think that they would serve three primary purposes:

- **Tricksters**: main goal is to mislead the player, or make their lives harder
- ** Helpers**: gives hints about any puzzles or locations where treasures might be or what items might help to beat the boss
- **Story Progressors**: more rare, story progressors can be found within dungeons and talking to them tells the database that the player is going to go down this certain path in the template, because now they have new knowledge that this NPC gave them
- **Merchant**: trades items with the player (in case what player picks up is useless to them) (might want to pepper these in sparingly, though; might lower the difficulty significantly if Merchant is in all or most of the levels.)

**Example Items**
- Standard potions that help with health
- Weapons that raise stats like strength
- Tablets that raise stats like health
- Magic scrolls that add new abilities
- Maps that reveal the map
- Story items that have to do with current quest or story arc
- Story items that may be able to be used in a later quest
- Scrolls with spells on them (fireball, teleport, heal, ect) (one time use or multiple?)

**World and Naming**

The world of MicroVentures is a fairly standard fantasy setting, but because of it's tongue-in-cheek humor and unpredictable story generation, the world has many elements that one would consider absurd. All sentient life is treated on the level of people, regardless of its appearance or feeding habits, and items from other dimensions are a common occurrence. The result is a mish-mash of cultures and technologies that should not logically work together, but somehow coexist in harmony.

Naming, like the other story elements, is handled by random selection from a list. The list will have a large variety of names, and one possible function outside of the MVP was pulling names from the player's contacts list and using them in the game.
Procedural Level Generation

Since it is a common element in Roguelikes, Microventures plans to utilized procedural level generation so that the player can experience an unlimited number of levels that do not get old. There are several dungeon generation algorithms that could be used for this project including the “Digging”, “BSP Tree” approach, and others.

For the minimum viable product, we will narrow our choices down to either either Digging or Binary Search Partition methods. It is a matter of tweaking either of these to create a generalized level that can match with the setting that is selected for the current adventure.

The “Digging” Algorithm

The “Digging” approach consists of starting with a dungeon full of wall tiles and a single “feature” in the center, where a feature may be a room, corridor, or other opening. The algorithm then enters a loop in which it selects one existing feature, and adds another feature onto it, slowly digging out into the dungeon. This guarantees that every room will be accessible, and while it seems to just start in the middle and expand outwards, it generally ends up looping back enough to form fairly interesting levels, albeit with some dead-end corridors.

Digging Algorithm Example

![Digging Algorithm Example](image-url)
Digging Algorithm Flow Chart

1. Fill Level With Walls
2. Acquire List of Possible Features (Eg. Rooms, Corridors)
3. Add A Feature In Center Of Level
4. Done Adding Features?
   - Yes → Done
   - No
     - Choose An Existing Feature In The Level
     - Choose A Wall Bordering The Feature
     - Choose A Type Of Feature To Add
     - Can Add This Feature At This Wall?
       - No
       - Yes → Add Feature
The “BSP Tree” Algorithm

The “BSP Tree” approach tries to plan out the general shape of the level in advance, rather than just digging out from the center. It works by partitioning the map into two pieces (along a random axis) and recursively continuing to split these pieces until the smallest ones reach a minimum size. Each split area becomes a child of the area it was split from. Once the recursion reaches its end state, a room is placed in each bottom-level area, and bottom-level children are joined by straight corridors. Then the program returns up the tree recursively and joins the children areas to their parents with corridors. This algorithm also guarantees that all rooms are connected, and it also prevents dead-end corridors from existing. However it will also result in very sparse levels with long corridors connecting distinctly-rectangular rooms.
BSP Tree Algorithm Flow Chart

1. Create Head of Tree Structure (Represents Entire Dungeon)
2. Are Leaves at Minimum Size?
   - Yes: Carve Out a Room Within Every Leaf Cell
   - No: Partition The Dungeon
3. Partition The Dungeon
   - For Each Leaf in Tree
     - Choose a Random Direction to Split Cell (Horizontal or Vertical)
     - Choose a Random Position (X position if Vertical Split, Y position if Horizontal Split)
     - Split The Cell at This Position (The resulting cells are children of the current cell)
4. Connect The Rooms
   - Start With The Lowest Level (The Leaves)
   - At Top Level?
     - Yes: DONE
     - No: For Each Pair of Cells at This Level
       - Connect With Corridor
Both the BSP and the Digging approaches have their benefits, and more advanced implementations may start with the “BSP Tree” approach and use the “Digging” approach to add more variety to and a more organic feel to the level, a direction we may end up going in depending on play-testing feedback. Currently we are experimenting with technical prototypes of both the “Digging” approach and the “BSP Tree” approach to dungeon generation.

Other Approaches
A third approach which might find some use depending on implementation and time allotment would be a combination of several maze generation algorithms modified to later change the maze to appear more sparse. Rather than having a level generated that has very thin walls like a maze, the algorithm will go about creating a much more rectangular-shaped maze and then blocking off chunks of the maze and inserting objects in certain locations. Since maze algorithms are a more generalized problem and consists of fewer abstract concepts, we can refactor some of the many maze generation algorithms to create a general outline of the dungeon that will then modify itself for doors, hallways, and rooms. Many maze algorithms such as Eller’s algorithm and recursive division have great efficiency in both runtime and memory usage, which make this option more attractive due to memory bottlenecks on mobile platforms.
Adding Content To Levels

A planned feature is, depending on the level theme selected by the story generator, the game will generate a level using different algorithms. As an example, the sparse levels generated through the digging method would be well suited for a cave setting. Conversely, the more organized mazes generated by a BSP approach would make a dungeon or a town level setting much easier to create, and the maze algorithms may be useful for more labyrinth-style dungeons.

The story generator will also determine which assets will populate the levels that are generated. If a boss is required for the story, then a boss room must be placed in the level and filled by the boss character. If the level takes place in a castle then it should be populated by guard enemies, while if it takes place in a forest it might be populated by mystical creatures. After the level generation algorithms discussed here generate the general layout of the levels, our engine will then go through the levels and populate them with enemies, items, and tiles that fit the generated theme and story for the level. Items can be placed at the end of dead-end corridors, and low-level enemies should spawn near the player’s starting location while more difficult enemies spawn closer to the boss or end of the level. Pathfinding will be used to determine where the starting location should be placed, which should be far away from the goal of the level.

Sources, Examples and Descriptions

General:

Digging Algorithm:
- [http://openprocessing.org/visuals/?visualID=18822](http://openprocessing.org/visuals/?visualID=18822)

BSP Tree Algorithm:
- [http://doryen.uptalys.net/articles/bsp-dungeon-generation/](http://doryen.uptalys.net/articles/bsp-dungeon-generation/)
- [http://donjon.bin.sh/dungeon/about/](http://donjon.bin.sh/dungeon/about/)
- [http://donjon.bin.sh/d20/dungeon/](http://donjon.bin.sh/d20/dungeon/)

Maze Implementation:

Level Morphing:
- [http://doryen.uptalys.net/articles/dungeon-morphing/](http://doryen.uptalys.net/articles/dungeon-morphing/)
Story Generation

Template Story Generation

The story generation in the MVP will be a phrasal template system at its core. The concept of a phrasal template is very simple: an idiomatic phrase is written, and then a number of words are removed, leaving open fields in their place. Then, new words that fulfill the same grammatical function as the words that were removed are inserted into the open fields, and a new phrase has been created. A popular implementation of phrasal templates is the word game “Mad Libs”.

Dinner’s great at ____________, but like a lot of good ____________, the (name of restaurant) (summer flings/single malts) place works best in the early afternoon, the wind off the Hudson rustling the

__________ in the back yard overgrown with ____________. The sound of (type of tree) (type of plant) ____________ a fine accompaniment to the ____________ and (something that makes noise) (bass notes/rough edges/peaks) ____________ of the chef’s more eccentric fare. But as Ethiopian pop (flat notes/smooth surfaces/valleys) music king Bizunesh Bekele once sang, “______________.” And every restaurant, even a new one, has got to start somewhere. For (name of restaurant) ____________, that somewhere, is now.

* For MicroVentures, the system keeps a story bank of its various actors (characters, enemies, items, etc.) and their states, such as relations to one another, alive/dead status, and such. When a story is generated by first assigning each main character an overall story arc, which is a large-scale goal the narrative will be built around. Then, story is generated on a case-by-case basis when the players generates a new game. Individual games are created by randomly selecting a template from a set of pre-written story templates. Each template will have fields for the main character, the environment, the types of enemies, and any other actors it needs, but their plots and goals are unique.

Once the game is completed, either by the goal being reached or by the player dying, the story bank updates its actors, mainly the main character who either died or gained new items.
One of the most notable implementations of phrasal templates in story generation was Michael Lebowitz’s Universe system. Universe was used to generate plot outlines via the selection of and expansion of plot fragments. A plot fragment was an abstraction of a set of goals and actions actors could take to meet those goals. Each contained an end state and a flow of action, a tree of scenes the system could choose from to create a plot to reach that goal.

To generate a plot, Universe would randomly select one of these fragments, and then move down its flow of actions, randomly selecting one action for each scene and trying to fill in the blanks of characters it needed by inserting characters from its story bank. If the character it tried to insert did not fit the profile of the blank, the selection was rejected and a different character was chosen. If it was unable to fill all of the roles for that action, a new action was chosen. If none of the actions could be performed, the plot fragment was rejected and a new one was chosen.

---

**Name:** LIZ CHANDLER (LIZ)

**Marriages:**
- DON CRAIG [DON] [AMF1] [1980]
- TONY DIMERA [TONY] [AMF3]

**IPRS:**
- HUSBAND-WIFE TONY DIMERA [TONY] 0/-8/-8/-8/-6/-6/-7/-3
- EX-SPOUSES DON CRAIG [DON] -5/-5/-4/-4/-0/-4/-4/-4

**Stereotypes:**
- ACTOR KNOCKOUT SOCIALITE PARTY-GOER

**Trait modifiers:**
- (SEX F) (AGE YA) (WEALTH 3) (PROMISCUITY-3) (INTELLIGENCE 3)

**Overall description:**
- WEALTH 8
- PROMISCUITY 3
- COMPETENCE NII.
- NICENESS 0
- SELF-CONF 6
- GUILLE 7
- NAIVETE 7
- MOODINESS 6
- PHYS-ATT 7
- INTELLIGENCE 7

**GOALS:** (FIND-HAPPINESS BECOME-FAMOUS MEET-FAMOUS-PEOPLE)

**AGE**
- YA

**SEX**
- F

---

**A typical UNIVERSE plot fragment.**

**PLOT FRAGMENT:** forced-marriage

**CHARACTERS:** ?him ?her ?husband ?parent

**CONSTRAINTS:**
- (has-husband ?her) (the husband character)
- (has-parent ?husband) (the parent character)
- (< trait-value ?parent niceness) - 5
- (female-adult ?her)
- (female-adult ?him)

**GOALS:** (churn ?him ?her) (prevent them from being happy)

**SUBGOALS:**
- (do-threaten ?parent ?her "forget it") (threaten ?her)
  - (dump-lover ?her ?him) (have ?her dump ?him)
  - (worry-about ?him) (have someone worry about ?him)
  - (together * ?him) (get ?him involved with someone else)
  - (eliminate ?parent) (get rid of ?parent (breaking threat))
  - (do-divorce ?husband ?her) (end the unhappy marriage)
  - (or (churn ?him ?her)) (either keep churning or)
  - (together ?her ?him) (try and get ?her and ?him back together)
Creating a simple plot outline.

* (tell '((churn neil liz))

working on goal - (CHURN NEIL LIZ)
Several plans to choose from FORCED-MARRIAGE LOVERS-FIGHT JOB-PROBLEM
- using plan FORCED-MARRIAGE

working on goal - DO-THREATEN STEPHANO LIZ "forget it"
- using plan THREATEN

⇒ STEPHANO threatens LIZ: "forget it"

working on goal - (DUMP-LOVER LIZ NEIL) - using plan BREAK-UP.

⇒ LIZ tells NEIL she doesn't love him

working on goal - (WORRY-ABOUT NEILL) - using plan BE-CONCERNED
Possible candidates - MARLENA JULIE DOUG ROMAN DON CHRIS KAYLA
Using MARLENA for WORRIER

⇒ MARLENA is worried about NEIL

working on goal - (TOGETHER NEIL)
Several plans to choose from SEDUCTION DRUNKEN-SNEAK-IN
SYMPATHETIC-UNION JOB-TOGETHER
- using plan SEDUCTION
Possible candidates - DAPHNE RENEE
Using DAPHNE for SEDUCER

⇒ DAPHNE seduces NEIL

working on goal - (ELIMINATE STEPHANO)
Several plans to choose from ATTEMPTED-MURDER EXPOSE
- using plan ATTEMPTED-MURDER
Possible candidates - RENEE ALEX
Using RENEE for KILLER

⇒ RENEE tries to kill STEPHANO

working on goal - (DO-DIVORCE TONY LIZ) - using plan DIVORCE

⇒ LIZ and TONY got divorced

working on goal - (TOGETHER LIZ NEIL)
no acceptable plans
The advantage of a template system is that variation is easily achieved and new templates can be added to the system relatively painlessly. However, the system is limited by the fact that games are more or less independent of each other - while elements from one can carry over from another, the actual templates and scenarios will never change in response to one another. Also, if not meticulously limited, templates can create strange and unorthodox stories. For example, “___ rescues ___ from ___” naturally lends itself to something like “Knight rescues princess from dragon”, but it could also just as easily end up as “Dragon rescues goblin from princess”.

Of course, with the tongue-in-cheek nature of MicroVenture’s story, this idiosyncrasy could actually be considered a plus, provided cases for strange combinations are carefully handled.

Minstrel Story Generation

Outside of our MVP, we want to implement Minstrel Story Generation as our means of creating story and level crafting in MicroVentures. Minstrel is a case-based story generating system that pursues not only character goals, but also story goals, thematic goals, as well as good presentation and consistency.

Case-based reasoning, or CBR, is a procedure that let’s us solve a problem through the use of previously-solved problems. First, we are given a problem. Rather than going directly after the problem, the system then searches in it’s memory for a similar problem that has already been solved. The system then adapts this past solution and uses it to assess a solution for the current problem.

Minstrel applies this problem solving model, CBR, to story generation. For an example, let’s say that so far in a story, we want to spice up the story by having a sad Knight who dislikes himself for whatever reason. Unfortunately, there is no action in the world for the instance of a Knight who dislikes himself, or at least the story generator doesn’t know what to say for a depressed Knight. By case-based reasoning, the system searches it’s own memory of a story but similar to our intended story direction. The system may pull up a scene where the Knight dislikes a troll, and therefore kills a troll. Though the action of killing is towards a troll in this instance, we can still adapt this situation since it involved the Knight disliking a subject and following it up with an action. Finally, we assess the solution by switching out the troll for the Knight, or the subject the Knight currently dislikes. The solution is now an instance of the Knight killing himself (the Knight), a very dramatic scene indeed but perfectly logically.

The way that Minstrel does this is through the use of TRAM. TRAM, or Transform Recall Adapt Method, is the systematic way that Minstrel is able to do the above: Take a currently existing problem, transform the current elements of the problem to match a previously solved problems,
then adapting the closest solution to the current problem. Here is an example of how TRAMs work below:

What’s nice about TRAM is that it can work recursively: if it doesn’t find a solution, it just transforms again recursively until it is able to find an adaptable solution. In the graph below, we can see how recursive creativity can make sure that our story never hits a wall.

![Diagram of Recursive Creativity]

*Figure 2.6 Recursive Creativity*
Unlike the Mad Libs approach, Minstrel does not rely on randomness to generate a story. Rather, Minstrel uses logic and available resources in everything it does, even in instances where the future of a story is resoundingly unknown. And even in the rare cases where Minstrel does use randomness, the system has already narrowed the resources enough where any of the random choices will be completely consistent and acceptable for the next part of a story.

Minstrel works for not only character drama, but also story goals as well. Story goals are essentially a step above character goals: While character goals govern the responses to their feelings, the author goals shapes the thematic and dramatic elements of the story. These author goals dictate the direction a story need to go in. Author goals are much more complex in nature since they deal with abstract story items, including things like theme, foreshadowing, motif, the structure and direction of a plot arc, the things that essentially make a story powerful and worth retelling. Therefore, the process of actually creating these author goals do require a level of planning. The graph below shows how we select from a pool of goals, then plan out for this goal to work within the story context by inserting scenes into the current story and recapturing exactly what was told.

Author goals use the same TRAM technique that character goals use, however it is much more difficult to implement TRAM for author goals. The reasoning for this is that author goals are simply hard to adapt due to them being so unique and varying in nature. Therefore, one of the backup of author goal generating, when the system simply can’t produce a unique author goal, is to provide generalized default versions of author plans that are guaranteed to work. For example, one of the default author goals, a consistent goal, is to make sure that all character goals and states remain consistent if the generator cannot provide a good enough author goal.

Procedural Storytelling in MicroVentures

For our MVP, we plan on using Template/Mad Libs version of our story generator. Our approach is the utilization of:

1. One Story Bank
2. Many Templates

Though the intricacy of these system can get complex, these are really the only things that are needed for the Mad Libs story generation.

The Story Bank essentially works as a storage of all possible story items that can come up in the game. This includes:

- Characters
- Enemies
- Important Items
- Locations

All of these items have attached to them states relative to their possible condition. For example, a Character can have states such as Health, Current Location, Kidnapped or Not Kidnapped, items they possess, and so on. There may also be certain importance or chance percentage attached to specialty items or locations as well.

The Templates are going to be what builds our story, or in other words, empty Mad Libs sheets. However, instead of people filling in the spots of these templates, the system attempts each of the templates it has onto the Story Bank. Once there is a template that matches the current state of the Story Bank, then it is used for the Adventure.

For an example of this, let’s say that the current state of the Story Bank states that there is a Warrior who is unconscious and a Wizard who is still alive. This current state might possibly match a Rescue-themed template, where a character in good health rescues another character who’s health is unconscious or worse.

Once the template gathers assets from the Story Bank, the game creates the story for the player to read over. It’s at this time the Story Bank’s assets for the current story is sent over to the Level Generator to create the level. These assets will determine who you play as, the environment tile set to use for the level, the final boss, the goal, as well as story quips and items during the game.

What’s great about having this system is that the Story Bank can easily be updated after each Adventure with accurate states for the story items that will change how the system uses them. At the end of each Adventure you play, the game will determine the outcome of your play-through, including the state of your character, enemies, items you’ve picked up, etc. This information is fed back into the Story Bank, which updates all the states of the Story Bank assets you interacted with.
By doing so, previous Adventures can have a tangible effect on subsequent ones and tie together into a cohesive plot. So going back to our earlier example of the unconscious Warrior, if you succeed with the Rescue Adventure using the Wizard, the Warrior’s state is now changed to healthy and the Template system will now be able to utilize templates where the Warrior is in a healthy state.

Here is a simple diagram of the Universe Generation flow for our game would look:

![Universe Generation Diagram]

To take template-style generation one step further, the implementation of a “boredom” system would protect against repeated adventures. In essence, “boredom” plays a part in story generation by checking for how frequently a particular template or story items has been used. As an item is used more often, the system weighs its choices away from that item - it is not removed, but its likelihood of showing up again are much slimmer.

Beyond our MVP, we want to implement Minstrel story generation for MicroVentures. Although we are still currently researching Minstrel and how exactly it works, one thing we do know is that Minstrel itself is a very complicated system, possibly too complicated for the amount of time we have for the 170 Series.

Hence, one of the possibilities we have is to implement Minstrel Remixed, a current graduate project by one of our TA’s, Brandon Tearse. Minstrel Remixed is a project that essentially follows the steps of Scott Turner’s strategy of creating a storytelling system.
Here is a description from Brandon of the projects:

**MINSTREL**

Over a decade ago, Scott R. Turner created MINSTREL, a program that could creatively manufacture cohesive stories without the need for human input. Although these stories contain diction akin to the writing of a child, they are nonetheless complex at many levels. MINSTREL was able to plan and achieve goals such as dramatic rise and fall, realistic human emotion, changing mental states, generation and followthrough with a final moral in mind, and generation of novel content. Unfortunately, MINSTREL is currently unavailable and was written in an old variant of LISP.

**Minstrel Remixed**

Minstrel Remixed is an attempt to recreate MINSTREL using Scala, a modern language that compiles to java byte code or translates directly to java source files. In addition to recreating MINSTREL, Minstrel Remixed attempts to surpass its predecessor in a handful of ways. Since Minstrel needs some sort of information to manipulate through its internal processes, it is limited by the size and variety of its internal libraries. We aim to allow users from the internet to generate information to enrich Minstrel's understanding of the world, allowing for rapid content generation and thus a quick growth in what Minstrel is able to come up with. In addition, since Minstrel Remixed will be running in real-time, it will be designed to accept a user-generated story modification and still come up with compelling stories despite the imposed changes. Lastly, Minstrel Remixed will be packaged into a library that can be plugged into other applications to provide realtime user-tolerant story generation.

We believe that system could work for MicroVentures story generating since:

1. It currently uses fantasy-style story generating, similar to the story structure in MicroVentures
2. The story in MicroVentures is simple enough for Minstrel to implement, and maybe even take it to another level with Author Level Planning, complex Character Goals, etc.
3. Even though Minstrel Remixed is in early stages and often gives weird/nonsensical stories, we believe this will only produce fun and absurd stories to play.
4. It can easily replace our Mad Libs style generation since it is essentially still producing story items for the level generator to work with.

One of the challenges we face with using Brandon’s system is the architecture itself: currently, this system runs in Java and Scala, while Marmalade runs on C++. As well, Minstrel Remixed is a very processing-intensive system, which might pose a problem even for modern smart phones that are barely starting to have dual core processors. One possible workaround is launching the engine off a server and having the game communicate with that server for stories. We are still brainstorming for solutions to these problems.
Story Integration in Game

Since it would be both boring and unreasonable to expect players to read a wall of text before each adventure and then present no story bits into a game, we want to introduce story throughout the entire MicroVentures experience. Here are some examples we would like to implement.

Opening

*Purpose:* Introduce overall quest: setting, character, enemy, goal.

Each time the player starts a new game, the story generator creates a new story based on the current story bank state. This story is displayed to the player as they enter the game and includes information for the player about the level they are entering. This information includes the location, the goal, and the end boss.

Instead of a wall of text, we want to split up the story into several pieces and display these pieces, with pictures, in the form of story book pages. We believe that this will make the story more palatable for the player, and the addition of pictures of story items will be a nice visual aid as well.
During Play

_Purpose:_ For when major events occur. Throughout the level, the player will trigger major events that affect the level such as unlocking a door, defeating a mini-boss, or finding a special item required to complete the goal. A story bit is displayed to the player when these events occur to differentiate them from normal actions. Some of these actions may be optional and affect the overall story regardless of the player’s success or failure in the level.

We are still experimenting with different styles and ways of displaying story for the player during gameplay, but two ideas that we came up with include:

1. Speech bubbles over the playing character during any of these minor events
2. Story text that fades in and out over the game, similar to the way it does in The Path or Swords & Sworcery

Closing

When a level ends, by the character dying or by completing the level goal, the result is displayed to the character in the form of a small story. When a player dies, the level goal is, of course, not completed so those goals are returned as failed to the story bank and this is conveyed to the player in the end story. If the goal is completed, the story bank will be updated and the player will be notified of their impact on the larger story arc.

This may be in the form of one single panel, since the outcome shouldn’t take long to explain.

To see a storyboard rendition of how both Universe and Minstrel story generation would work for the game, go to the Storyboard section on Page 57.
Art Style

Visual Style and Look
The art style in MicroVentures is heavily influenced by old school RPGs and even modern games that utilize pixel art such as Swords & Sworcery and the Bit Trip series. Though the game will have roguelike gameplay, we want to move away from the old school ASCII look of roguelikes and move towards a more stylized look to our game.

Below are several pages of reference art and prototype art for our game:

Swords & Sworcery

![Swords & Sworcery](image1)

Bit Trip Series

![Bit Trip Series](image2)
Art from Prototype

MicroVentures Reference Sheet
Sprite Sheet
The art content will most likely exist on a sprite sheet like the ones below.

MicroVentures Reference Sheet
Rendering, File Size Management, Sprite Dimensions

After further researching the Marmalade SDK about rendering there are a number of ways to draw the art within our game. Marmalade either uses its own rendering system, or OpenGL ES 1.0 or OpenGL ES 2.0. This is important to note because it affects the art style.

With OpenGL ES, image dimensions need to be powers of 2 to guarantee that it will be binded as a texture correctly for mobile devices, or else there might be the issue of the image appearing as a blank white box on some devices. We are aiming to use OpenGL ES for rendering as it is hardware accelerated and will run more efficiently on the phone. Due to the powers of 2 constraint, we are aiming to have sprites such as main characters, small monsters, items, walls, floors, etc as 64 pixel by 64 pixel images. For larger game pieces such as bosses we can go up to 128 by 128.

We have chosen these pixel dimensions for a number of reasons; the first being that since our artwork revolves around an 8-bit style, we don’t need extensive detail or extra space. The second reason is that we think this size will keep the total file size of the game down, and also keep resource loading quick when playing the game. Please refer to the PLATFORM section for more reasons as to why we are keeping our dimensions this size.

Animations and Sprite Asset Folder Layout

Animations in the game will be kept to a minimum, as the style of movement and gameplay is mean for quick play. Most animations in the game such as the main characters walking, idling, attacking, dieing, enemy movement, etc will be around 2-5 frames. Keeping in mind the dimensions sprites have to be, we will not be using sprite sheets when loading files into the game as having animations with a frame number not equal to a non-power of two can lead to memory waste; i.e. an animation of 3 frames will require a sheet that is 64 x 256 pixels which leaves one blank 64 x 64 square unused and taking extra memory.

To avoid this problem, we will organize our sprite assets folder in a unique way that follows:

- The sprite assets folder will be located in the assets folder (which can also have music, sound effects, level template data, tilesets, background images, etc)
- Inside the sprites folder, animations will have their own separate folders, with the name of the individual folder referring to the reference of that animation in the code
  - Inside each animations folder, there will be sprites of 64 x 64 (or whatever dimension) that represent each frame in the animation, and the name of each image will be 0, 1, ..., n-1, n. The number will refer to the order in which the frames should be organized to animate. The resource loader will manage this at loading time to save the binded texture into an array in which the frame order will refer to the order each image is numbered in the folder; upon playing the game, animation can be done by having the object’s draw function point to the different texture index in the array.
• Inside the sprites folder, non-animated sprites will not have separate folders like animations do, but instead they will be put directly into the sprites folder. The name of the file will be the same as the asset reference in the code.

Here is an example image for the sprite assets folder.

Our file type for artwork will be saved in the Portable Network Graphics (PNG) format, to allow for alpha channels in the game. This allows us to keep the 8-bit style while having another level of depth to the look and feel. An example would be that slime monsters will now be see through. Through research we also know that we can use OpenGL’s blend/alpha function to make use of these alpha channels.

**Backgrounds, Concept Art, Tile Sets**
We may possibly have separate folders for background images, concept art, and tile sets. The dimensions of these still need to follow the powers of 2 rule, but can be whatever size needed. We are still not sure if we are going to use tile sets for the game yet or simply have all the scenery art in the assets > sprites folder. If we have tile sets as one or a few images, we will do UV mapping to use a specific part of the tile set when drawing in the game.

**Particle Effects**
For some special effects in the game we will be using particle systems, either decorated with 16 x 16 image textures, or simple polygon shapes. Particle systems add to the look of the game which might be hard to do with simple animation for things such as a dragon blowing fire, water particles, slime monster goo explosions, etc.

**Custom Font (Nice To Have)**
We also want to have custom font in the game. This font may either be made by our artists, or if we can get copyright to already made fonts we will use them. If the artists were to make fonts Marmalade has Rich Font Support to help turn our bitmap fonts into a usable format.

**Resources:**
- [http://stackoverflow.com/questions/1928991/minimum-sensible-button-size-on-iphone](http://stackoverflow.com/questions/1928991/minimum-sensible-button-size-on-iphone)
Sound Style

The midi-fantasy style score will work alongside the art and level design to help establish the atmosphere of the game world and thus help define the tone for the gameplay. There will be a unique, short, loopable theme for each level. For the MVP this means a dungeon, forest, snow, and Hell theme. Each one will enhance the gameplay experience with its careful choice of instrumentation and harmonic structure.

GENERAL SOUNDTRACK
The dungeon theme will contribute coldness and creepiness to the tone by using midieval instrumentation and minor counterpoint. Tubular bells, pizzicato strings, harpsichord, a reed instrument such as oboe or bassoon, and perhaps a resonant wind such as a stone or clay ocarina could all help set the appropriate tone. Attention to space will be important to keeping a "creeping" effect.

The forest theme will also develop a creepy tone but with a more earthy and mysterious flavor. The melody could switch between a clarinet or other woody reed and a woodwind such flute, allowing soft human-like articulation. This, along with plucked strings, fretless bass, an unobtrusive lute or choir to fill in harmonies, and hand percussion such as tambourine and mallet drums will achieve a wondrous fantasy soundscape. I find that this magic property is more easily achieved with nonstandard chord tones, so this theme will most likely modal instead of simply major or minor.

The snow theme will convey a softer, less tense mood. It will use a choir, harp, prominent bells, modal piano arpeggios, and possibly warm horns or strings.

The Hell theme will set a tense, heavy and evil tone by utilizing dissonant harmonies and repetition of melodic motifs. A simple ostinato, perhaps a stepwise minor descent on a woodwind or strings, could be melodized over by hot brass instruments occupying different ranges to help achieve this mood. Occasional use of the tritone should be considered.

SOUND EFFECTS
These will be either created or chosen from free sources carefully to enhance the player's experience without distracting from it. There will be game world sounds, such as the warrior's sword striking or an enemy grunting, as well the sounds of the game itself, such as a click to proceed to the next line of text in a story.

SOUNDTRACK BEYOND MVP
We have ideas for several other environments we could create if time allows. These include mountains, desert, swamp, and the ruins of a town, and each would have its own unique theme or at least a unique variation of the main theme.
It would also be nice to have a theme for each main character. Whenever a new story is being told to the player, the Warrior's, Wizard's, or Rogue's theme would play. These would be short; the player will only listen to them while reading the story at the level's intro and this is presumably a much shorter amount of time than the time it takes to complete a level.

Finally, if there is a ridiculous preponderance of time, I would like to explore the three main characters' themes in greater detail in a title screen overture. The challenge this task presents is that the three character themes would then have to be easily inter-relatable. One solution could be to use a literal reference. For example, include in each theme a variation of a similar melodic or rhythmic motif, perhaps using different instruments. Or there is the slightly more subtle approach of harmonizing a single melody in a distinctly different way for each theme. This is something to be decided later if we come across the issue.

MAIN CHALLENGE
A similar challenge to the overall soundtrack will be the relation of each level theme to another. Each should make sense on its own, yet they all must help define each other. That is, they should all share certain stylistic qualities so that they sound like they belong to the same game. If this is not done carefully there will be a disruption in fluidity of gameplay experience.

We will employ a variety of techniques to help achieve this goal. The score will use only midi sounds of contemporary instruments as opposed to electronic synths or live recordings. It may also utilize recurring melodic and rhythmic motifs. Perhaps there can be one recognizable MicroVentures riff that is distinct from the character riffs. Harmonic relation between themes should also be considered. For instance, the level themes may be in distinctly different keys from each other or use different scales, while the character themes may all share a common key. These techniques will keep a common tie between the different themes while still maintaining each theme's individuality.
User Interface

Team Logo Screen / Splash Screen
- Team Name
The Team Logo Screen is the first screen that will be viewed by the player upon turning on the game, and will last from anywhere between ½ to 1 second. The purpose of this screen is for our team to take credit for the game the player is about to play by showing the team name. We may also show sponsors on this splash screen if we have any.

Title Screen
- Title
- Push to start Adventure
The Title Screen follows the Team Logo / Splash Screen. This screen is mainly here for aesthetic reasons. It informs or reminds the player the name of the game they are playing. It also gives first time players a first impression of the game. The Title Screen is designed to be an exciting invitation to play the game. It waits for user input and has text that reads “Press anywhere to start”. There will be background concept art that represents the game.

Main Menu
- Start Adventure/Continue Adventure
- Records (Check up on your story)
- Options
- Exit
The Main Menu Screen gives the player the standard options found in any game. You can start or resume your adventure from where you last left off. You can view your records which showcases miscellaneous items or accomplishments to the player. And you have an options button for player’s to customize the experience to their liking.

Records
- Stories Generated
- Music
- Credits
- Achievements (nice to have)
The Records Screen lets the player view miscellaneous data. You can pick “Stories Generated” which allows the player to view all stories generated from playing through their adventures as one large combined story, or the individual stories alone. This includes stories generated from past adventures that are no longer accessible after the player has reset the adventure. Any music composed from our sound artist will be accessible through the “Music” option. The “Credits” plays the credits showcasing all of those who have contributed to the project. Achievements is a nice to have in which we have unlockables, or goals in the game the player can seek to accomplish.
The Game Screen is the screen players will be spending most of their time on. The design philosophy behind this user interface is simple and accessible. Buttons on the screen are kept to a minimum in order to keep it clutter free.

As shown above the health is displayed as a red bar on the right side of the screen. It's relatively large because pressing on the bar, the player will heal their health if they have health potions in their inventory. This allows players to easily heal themselves, an action we expect to happen often during gameplay and therefore have made it extremely accessible to the player. To supplement the information given by the health bar, a number representation of health remaining is shown just above the health bar. This allows players to make meaningful decisions when attacking opponents, knowing how much damage an enemy can inflict on the player. Just above the health is the ad or attack power of the player. This also gives players a better understanding of how to tackle their enemies.

The option button is located at the bottom of the screen, providing access to a standard options menu.

The item button is located next to the options button. Pressing on this button will reveal an inventory wheel where players can swipe through the wheel to pick an item to use. Pressing on the button again will hide the inventory wheel, reducing clutter on the screen. The wheel shows four items at the time. The player can hold an unlimited amount of items and therefore the wheel will hold more items than what seems physically possible.

In order to use the items in his inventory, the player can tap the items in the wheel. If it’s an item that can be used on the player, then the item will be instantly used. If the item must be used on an enemy then the player must then click on the enemy to use it. Players can deselect an item by clicking on it again or tap the item button to close the wheel, and deselect the item. To scroll through the wheel the player holds on the screen and drags the wheel left or right. This setup allows the player access to a large inventory, while still playing on the same screen, allowing a much better experience.

In the top left of a screen is the adventure book. This book will open up a separate menu that describes the player’s current story bank as well as current adventure goal. When opened, this
takes up the entire screen to make reading easier and it’s something we don’t expect the player to do often. It will lay out the current micro story to the player telling what they accomplished in the session.

Upon using the Lens of Truth, the player will be able to see enemy attack and health statistics. This is displayed on the bottom left (attack) and bottom right (health) of every enemy unit the player runs across. This allows players to make better choices regarding which monster to do battle with.

Movement will be done by clicking on floor sections in the screen. Upon clicking, the player’s character will move quickly in an animated fashion to the designated spot. If a monster is on the path to the destination the player will be forced into battle, or they will pick up items if crossing them. Path finding will be needed to move the character around the level quickly and correctly. Upon first playing the game unexplored areas will be covered in black so that the player will need to navigate the area to reveal the world. The black will disappear or get more transparent if it is 2 or 3 blocks away, although this will need to be play tested.

Another way to view the world would be through the camera. By default the camera tries to center the player on the screen, unless they are near the border of the screen. With the grid size of 64 x 64 px blocks and a phone size of 960 x 640 px there will be a viewable area of 15 x 10 blocks of the world. The player has the option of zooming the camera by pinching, or moving the camera by putting 2 fingers on the screen and dragging in parallel directions. The camera will bounce back to the character upon releasing the fingers.

Chat bubbles appear during the story sequences of every adventure. To remove the chat bubbles and proceed with the dialogue, the player simply has to click anywhere on the screen. This is the best way to craft a seamless roguelike experience with story elements. The chat bubbles will be relatively small compared to a separate text box appearing for every spoken dialogue. This will force in game conversations to be minimal and simplistic, encouraging players to read through the dialogue instead of skipping past the blocks of text. This also makes it easy to understand which character is speaking at a given time, and removing the need to create separate art of the character if using a box dialogue like traditional RPGs.
Shop Menu - (Extra Feature?)

- Items
- Gold
- Combination Gems
- Continue Adventure
- Buy

Though outside the scope of our MVP, we might implement a Stop feature where the player uses gold picked up during gameplay to purchase items for use in the game, or possibly for story reasons. This feature is a relatively new idea and it is still being discussed.

Menu/Options

- Vibration
- Volume
- Music
- Clear story data
- Exit

The Options Menu allows the player to switch the Vibration on/off. Players can also toggle the music in the game so it will or will not be played. The clear story data clears the story database of everything that has happened in the game, essentially making it so that the game would be the same as if though first played.
User Interface Flow Chart Draft
UI Drafts
Physical Prototype

Story Generation
Before the players start the physical prototype, a story is generated using Widerule. The Widerule program introduces the player to the setting of Microventures. Widerule also picks one of the three generic characters (Warrior, Wizard, or Rogue) the character will play as, one of the two goal items that they are after (Princess or Treasure), and one of the two final bosses guarding the goal item (Dragon or Slime Slug). It also gives the character a perk depending on what happened during his adventure there. Since the player hasn’t done a previous adventure, it randomly chooses one.

Preparing the Game World
Once the story is created, the game board is made. Since it’s a physical game board, we randomly choose wall tiles, enemies, and chest to put randomly put them on the board. However, some of the items on the board are not procedurally generated. The player’s character, final boss, goal item, final boss room, and key to the final boss room are static variables. The final boss room is randomly put on the board somewhere in the top left corner of the board with the final boss and goal item in it. The final boss is put at the entrance to prevent the player from reaching the goal item before defeating it. The player, on the other hand, is put into the bottom right corner. The chest with the key in it is put somewhere near the boss. Everything else is randomly put on the board in a way that the player can move freely anywhere on the board.

Player Movement
The player could only move one tile per turn. They can move to any adjacent space horizontally or vertically that isn’t blocked by a wall tile. When a player is two squares horizontally, vertically, or diagonally away from a monster, that monster moves towards the player using the same rules a player moves. The player can move onto a monster tile and monsters can move onto the player tile. The player can also move onto a chest to receive an item. The player can move onto the goal item to finish the game.

Combat
Each character has two statistic integers, health and attack, that are used during combat. When a player moves onto a monster tile or a monster move onto the player tile, combat is initiated. If the player moved onto the monster tile, they attack first. Otherwise, the monster attacks before the player. When attacking, a random number between one and the character’s attack variable is selected using a dice. That number is then subtracted from the other character’s health. If one of the character’s health variables reaches 0 or below, that character dies and is removed from the board. If both character have attacked once and both their health variables are greater than 0, the advancing character move back to its previous tile. The player ends the game when his health variable is 0 or lower.
What We Learned

From the physical prototype, we learned:

Pros
- Fast-paced action is fun
- Procedural Storytelling and level generating makes for fun game-play
- Items make up for minimum stats

Cons
- Needs more items
- Needs more varying stories and better ways of telling stories.

Conclusion
Our physical prototype was fun! It was easy gameplay, and the procedural aspect of the levels made each play through unique and challenging. What we lacked the most, however, was more items for variation in gameplay. Players also seemed to desire a better way of storytelling through the game rather than just a wall of text right before you play.
Virtual Prototype

Gameplay
Our virtual prototype, created in Game Maker 7, focused much more on the gameplay. The main influence of our virtual prototype was of course Desktop Dungeons, which we adapted into the prototype. The only differences were that stat-building in the game depending on simple item pick-ups as well as the actual time elapse per level you played.

We built each level (3 levels total) to estimate around 2-4 minutes long, with two levels of enemies and a final boss at the end. One of the other key objectives for this prototype was to test out the balance in the gameplay. With a game as small as ours, it is very easy to shift the game from too easy to impossible. Therefore, we tried mapping out exactly where the varying enemies should be, as well as the amount of items we can give the player.

The player movement was using the arrow keys, and attacking was done by running into an enemy. Your health and attack stats are in the form of numbers right next to your character.

We were unable to implement a random level generator in time for this prototype. However, by using a static board, we can get a good model of how our levels should be, which in turn will help in the crafting of the Level Generator for the final game.

Story
We were not able to implement a procedural story generator either, mainly since couldn’t procedurally craft a level out of the created story and we because already have a simple WideRuled story generator from the physical prototype.

However, what did emulate were the different directions a story could go and the gameplay that would follow. In our prototype, we had a very standard story where you play as a Warrior rescuing a princess from a Dragon. If you succeeded, then the next adventure was about the Wizard who was told by the princess about her overhearing about an ancient relic, in which you quest in the forest to look for it. If you died, then the story would be with the Rogue who was hired by the King after the Warrior’s failure.
What We Learned

From our Virtual Prototype, we learned:

Pros

- Roguelike gameplay is fun
- Story forking
- Relatively balanced after tweaking

Cons

- Wall of text is too much – need better way of portraying story.

Conclusion

Players enjoyed the gameplay, and some naturally compared it to Desktop Dungeons. They liked how fluid the gameplay was and the item pickups in the game. They also enjoyed the forking of the story at the end of the first level. However, similar to the first physical prototype, we ran into the same problem where the wall of text at the beginning was too much for the player.
Once upon a time, … (Show the beginning text)

There was a young boy. He and his friend were very bored. His friend suggested that they should play ball and that he should go get the ball. The young boy agreed and went to go grab the ball. (move the character over the ball)

The young boy picked up the ball and threw it to his friend. His friend then threw the ball past the young boy. The young boy’s friend yelled to him to go grab the ball. Disappointed, the young boy went to get the ball near the tall tree. (move the character to the tree)

As the young boy grabbed the ball, a sword fell from the tree right in front of him.
Filled with curiosity, the young boy picked up the sword and examined it.

Just then, a dragon flew above his city, burning everything in sight. The young boy raised his sword and charged into battle. (move the character over the dragon)

A mighty blow was struck upon the dragon, but just enough to damage him slightly. The dragon reached to grab the young boy. With his lack of experience, the young boy was unable to dodge and was tossed into a nearby building.

The Dragon then grabbed the princess and flew off into the mountains. As the young boy gained conciseness, a random commoner stared in awe at the sword the young boy possessed.
He exclaimed “The sword of the divine, the hero has been chosen!” Villagers from all around gathered and talked amongst themselves. The king parted his way through the crowd and assigned the young boy the privilege to rescue his daughter. The crowd cheered as they push the young boy out of town towards the dragon infested mountains.

(story prologue)

The young boy sees a hostile slime slug ahead as he makes his way to rescuing the princess. Time to show what he’s got! You are told to tap the slime slug.

The young boy approach the slime slug and strike him with your sword. And victorious you are in defeating the slime slug!

After his defeat, the slime slug seemed to drop a useful item: A health potion!
The young boy puts the health potion in his inventory for later use. He is also introduced to his item wheel menu.

There was a sign posted at the edge of the town. It was too hard to read and the young boy tried to rip it apart in anger. (use zoom in feature on phone)

Trying to crush the board only made the text unreadable again. The young boy stormed away and headed off to save the princess. (move the character to dungeon entrance)

The young boy basked in the glory of defeating the slime slug, but the way was still covered with fog. Without hesitation, the boy rushed into the darkness. (move the character toward the clouds to revile more tiles)

Along the way, the boy discovers a chest off to the side. His curiosity leads him to open it and see what’s inside. (move over the chest)
The young boy scavenges though the chest and finds a key and map. He holds the map up high to remove the cloudy darkness. With the dragon in site, the young boy pushes forward toward it. In his way stands a skeleton warrior, forbidding him from reaching the dragon. The young boy smites it with his mighty blade. (move the character over the skeleton)

A powerful strike was lain upon the skeleton, but not enough to finish it. The skeleton raises his blade and strikes the boy with a powerful slash. The boy feels pain and feels that he might lose this battle. (show the health stat)

Within that brief second, he remembers the potion of health he received from the slime slug. (select the items tab and select health potion)

Revived from his wounds, the young boy strikes the skeleton warrior again. (move the character over the skeleton...again)

The skeleton crumbles to dust as soon as the blade hits it. As the skeleton disappears, a powerful set of armor takes its place. The young boy takes and dons the armor. (move the character over the armor, select the items tab, and select the armor)

The young boy feels increasingly powerful once he don the armor. He feels mighty enough to slay any dragon. (show the attack and health modifier increase dramatically)
With nothing in his way, the young boy races towards the dragon. (move the character towards the dragon)

The young boy raises his sword and strikes the dragon. (move the character over the dragon)

The dragon takes a considerable amount of damage. The Dragon then strikes the young boy with all his might. (Show the health lower) The young boy flies back in pain. He is wounded, but shows a great passion to go on. He raises his blade again and strikes the dragon in its underbelly.

“NO! I HAVE BEEN SLAIN!” the dragon yells as he falls and disappears in front of the young boy eyes. “My hero!” the princess exclaims as she come running up from behind the dragon. “Now take me away in your arms.” As the young boy notices the princess, he also spots the dragon’s treasure behind her. The young boy thinks for a bit and makes his choice to take the princess or the treasure. (move the character over the princess or the treasure to end the game)

The End
Adventure Storyboard
This is a storyboard of one typical Adventure playthrough and several forking possibilities.

1. The menu is shown at start up.

2. The player clicks “Start” and a generated story is displayed.

3. The player starts off seeing just the princess, some walls, and a fog-of-war which obscures the rest of the level.

4. The player touches the screen where they want the princess to move.
5. As the princess is moved toward the fog-of-war, the fog disperses, revealing more of the level.

6. As the princess explores the level, a slime enemy appears.

7. Every time the princess moves, the slime also moves.

8. When the player clicks on the slime, the slime’s health and attack values are displayed around it.
9. When the slime is selected, the 'Attack' button is shown in the actions menu.

10. When the player clicks on the 'Attack' button, the princess and the slime attack each other, dealing damage based on their attack value.

11. When the slime is defeated, a chest appears where the slime used to be.

12. When the player clicks the chest, a potion appears above the chest.
13. After clicking the potion, the potion is placed in the inventory menu.

14. When the player clicks the potion, the potion heals some of the princess' health lost from the fight with the slime.

15. As the princess explores, she comes across a large chest.

16. When the player clicks on the chest, a key appears above the chest.
17. After the player clicks the key, the key is placed in the inventory menu.

18. Since picking up the key was a major event, the information tab lights up indicating a new story point.

19. When the player clicks the information tab, it reveals new story information.

20. As the player continues to explore the level, they find the door to the dragon’s lair.
21. When the player clicks the key, the door to the dragon’s lair is unlocked and the key is removed from the inventory menu.

22. Unlocking the door is a new major event and the information tab includes a new story point.

23. The princess is moved into the dragon’s lair and the player notices the dragon and captured knight inside.

24. After engaging the dragon in combat, the princess defeats the dragon.
25. With the dragon out of the way, the princess may now reach the knight. Clicking on the knight completes the quest.

26. The player is shown a success screen with their accomplishments summarized in a small story.

27. The player is returned to the main menu after reading the success screen.

28. After starting a new adventure, the player is shown a new story screen that has been generated using the player accomplishments from the previous level.
29. It is also possible that the princess dies to the dragon.

30. The player is shown a failure screen describing what happened in the form of a small story.

31. The player is returned to the main menu after reading the failure screen.

32. After starting a new adventure, the player is shown a new story screen that has been generated using the player’s accomplishments.
Player starts MicroVentures. He didn’t leave in the middle of a previous game session, so a new adventure is needed.

**Step 1: Choose playable character**
The system accesses the story bank and chooses one of the main heroes that is still alive - in this case, The Warrior.

**Step 2: Check character’s previous adventures**
The system consults the story arc related to the chosen character and notes what kinds of adventures have already taken place, how they ended, and how far from the end of the arc the plot is.

**Step 3: Choose story template**
The system parses the collection of story templates that The Warrior can participate in, weighting its choice in favor of templates that haven’t been used in the story arc yet. It chooses the “CAPTURE” template.

The story will be something along the lines of:
“The Warrior must catch <target>, which is in <location>.”
Step 4: Fill in the blanks
The system begins to fill in the fields of the “CAPTURE” template. For the goal of “catch ___”, the system can use either characters from the story bank or monsters from its assets.
Oh my, it chooses one of the other heroes, The Rogue!

It checks the environments associated with The Rogue, and chooses “City” from the list.

It checks its enemy types assets and looks for a few that are allowed in that environment. It chooses the “Thief” and “Rats” types.

It checks its item assets for ones to populate the environment with and chooses items The Warrior can use.
Step 5: Generate a story
Now the system knows what assets and characters are involved in this particular story. It generates a map based on the environment, enemy, and item choices it made, and a story to give the player.

“The King has sent The Warrior to capture The Rogue, a villainous scumbag living in the city. The coward is sure to run, but his thief and rat minions will undoubtedly try to help him escape.”
## Minstrel Storyboard

<table>
<thead>
<tr>
<th>STARTING STATE</th>
<th>TRAM 1</th>
<th>ADVENTURE 1</th>
<th>OUTCOME 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Knight</strong></td>
<td><strong>Knight</strong></td>
<td><strong>You are the Knight, and you are on a quest to</strong></td>
<td><strong>Knight Fails</strong></td>
</tr>
<tr>
<td>- Disdain towards Dragon</td>
<td>- Transform</td>
<td><strong>defeat the dreaded Dragon and get back the</strong></td>
<td><strong>Knight is Dead</strong></td>
</tr>
<tr>
<td><strong>Dragon</strong></td>
<td><strong>Knight Recall</strong></td>
<td><strong>King’s Jewels.</strong></td>
<td><strong>King’s Treasure stolen</strong></td>
</tr>
<tr>
<td>- Stole King’s Treasure</td>
<td>- Wizard held disdain</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- towards Slime Slug</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Knight Adapt</strong></td>
<td><strong>Knight Adapt</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Disdain towards Dragon</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>State 2</strong></td>
<td><strong>State 2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Knight</strong></td>
<td><strong>Author Goal</strong></td>
<td><strong>Author Recall</strong></td>
<td><strong>Author Revives Knight</strong></td>
</tr>
<tr>
<td>- Dead</td>
<td>- Bring Knight back to Life</td>
<td>- Brought Rogue back to life.</td>
<td><strong>Knight is back to Health</strong></td>
</tr>
<tr>
<td><strong>Dragon</strong></td>
<td></td>
<td>- Wizard revitalized Rogue’s corpse</td>
<td></td>
</tr>
<tr>
<td>- Still has King’s Treasure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Author Goal</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Bring Knight back to Life</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Author Adapt:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Wizard revitalize Knight’s corpse</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You are the Knight, and you are on a quest to defeat the dreaded Dragon and get back the King’s Jewels.

You are the Wizard, on a special quest to find the Knight’s body. You only hope that it’s not too late...

Wizard Revives Knight

King’s Treasure stolen

Knight is back to Health
Letters of Commitment

November 2, 2011
I, Eric Lawson, commit to be the composer and sound effects designer for CS 170-172 project Microventures. I will meet regularly (at least once every 1-2 weeks) with the team to stay updated with their progress and to go over mine. I will help to round out the player's experience via the soundtrack and can dedicate at least 5 hrs a week to compose, record, produce, and mix/master the sound effects.

November 5, 2011
I, Elmer Zhu, commit to dedicating 12-15 hours per week in providing art assets for the group making the game Microventures.

November 6, 2011
I, Xian Chua, Pledge to devote 12-15 hours a week on Microventures as a Game artist.

November 5, 2011
Brandon Tearse
As your TA, I'm committed to helping your team to get whatever materials and information you need to help your game thrive. If that's information and source behind Minstrel, networking, or other help, I'm more than happy to provide it whenever you need for the remainder of the sequence.

November 7, 2011
I, Lauren E. Scott, commit to putting in at least 5 hours of writing/authoring work per week for the game Microventures, barring any unforeseen circumstances. This letter of commitment carries over for the 171 and 172 classes, and this agreement can be terminated by the Microventures team as they see fit.
Conclusion

We are all extremely excited in pursuing MicroVentures as our final game project for the CMPS 170 series at UC Santa Cruz. With the utilization of both mobile platform development and procedural level/story generation, we know that the coming months will be one of the hardest challenges in our college career. We also know that though these challenges may seem tough right now, we are engineering students who have survived UCSC’s Computer Science course (which includes Mackey), and have the tools and know-how to get the job done!

MicroVentures is huge step for all of us, and will require dedication and teamwork to a degree that we have never been exposed to before. But we’re ready for the challenge, and we’re ready to create something new, innovative, and most of all, something we can proudly show the world.