Game Design Document (GDD)

Letter Storm
GG Productions

Version 1.0 Created: October 16th, 2014

Last Updated: October 16th, 2014

Table of Contents:

1: Executive Summary

1.1 About this Document

This document describes the game design of the Project: Letter Storm. This is a living document and can be changed at any time at the discretion of the developers.

1.2 Motivation

It is well known that if a child does not find learning software fun to use they simply will not use it. In this sense, software that advertises itself as educational yet is panned by its target audience is somewhat paradoxical, for you cannot teach someone if they are not paying attention. Thus it can be concluded that, for educational software to be successful, it must engage children over the long-term while simultaneously teaching them. Our team has set out to develop a product that meets these requirements: Letter Storm, a game that is both entertaining and educational.

1.3 High Concept

Letter Storm is an educational top-down shooter PC game that teaches you how to spell by challenging you to defeat monsters to collect letters, and using those letters to spell out and defeat bosses.

1.4 Core Gameplay

Letter Storm's core gameplay is that of a 2D shooter game. Players must proceed through the game without losing all of their given set of finite lives or health. To defend themselves, players can shoot at monsters in the shape of alphabetical letters. Upon being defeated, these monsters turn into collectible items in which the player can collect into his or her inventory. The collected letters can then be used at the boss stage, in which the player must use to "shoot the letters" to "spell" and defeat the boss at the end of the stage. Other elements have been incorporated into our design. They include elements such as obstacles, power ups, and a running score. A user can also input a customizable word list in order to challenge a player in spelling specific words.

1.5 Genre

Letter Storm is categorized as a Shoot 'em Up, Top-down Shooter, and (Spelling) Educational PC platform game.

1.6 Target Audience

While Letter Storm can be used by anyone, the target audience is early elementary school students. The intended audience for Letter Storm are children, both boys and girls, between ages 4 to 8 for gameplay, and educators and parents looking to teach their students or children how to spell.

1.7 Team Members / Jobs / Contact Info

Team Member	**Role**	**Contact**
Nabil Lamriben	Project Leader / Design Leader / Environment & integration Leader	lamribenn@gmail.com
David Lustig	Backup Project Leader / Implementation Leader	davelustig@hotmail.com
James Raygor	QA Leader	jamraygo@bu.edu
Jeannie Trinh	Requirement Leader	jtrinh@bu.edu
Paul Pollack	Design Leader/ Configuration Leader	pdpman@bu.edu

2: Game Overview – David/Nabil

2.1 Overview

The main character, Albert or Annie, has discovered a lab overrun by Character Monsters. Higher motivations drive him to attempt to rid the area of these creatures, collecting the letters associated with them along the way. During his journey, he will encounter large creatures that can only be defeated by spelling a word correctly using the letters collected. In this way, the user will learn how to spell while enjoying the gameplay elements of a 2D scrolling shooter.

2.2 Story

It was a dark and stormy night, and two siblings, Albert and Annie, were caught out in it. As the wind howled and the trees creaked, they plodded through the nearly blinding rain toward a hope of refuge in the distance. The dark, rectangular shape on the horizon would have been foreboding under any other circumstance, but to them it symbolized shelter and warmth. With each passing minute, hopes were strengthened as the structure slowly grew larger and clearer.

Then, their moment finally came. After journeying for what seemed to be hours, the two found themselves before the large, unsecured iron gates of what appeared to be an old college building of sorts. It took only one look at the rusted iron bars and crumbling brick walls to see that time was no friend to this work of man. Yet, they were not deterred. Perhaps they could take temporary shelter inside until the storm passed, or at least such was the thought.

Slowly they worked their way through the gate and to the facility's oversized engraved doorway. Whatever fantasies of danger it would have stirred up under normal circumstances fell prey to the desperation of their plight, and so, without much thought, they eagerly pressed the two iron-clad doors open. Their entrance did not go unannounced, though, as the age-old hinges let out a moan strong enough to shake one's bones. And that is when the trouble started.

Looking into the dark interior, one could make out a large foyer walled in by deep hallways and a large, central staircase. The faint but growing noise similar to that of frantic footsteps emanated from the core of the structure. Then they saw him: A scholarly gentleman with his hair amuck and his clothes wrinkled and stained came running madly down the staircase right toward them. "Help! Help!" he yelled, gasping for breath between his words. "Something has gone terribly wrong. The sound of your entry startled me into a grave accident. My lab is now run amuck with creatures not from this world."

At the sound of such news, the two siblings began to feel fear well up within them for the first time that night. "It's alright to be afraid, but do not let fear stop you from doing what is right. I have here the book that the creatures came from and the pencil used to design them. If you use them correctly, you should be able to vanquish the creatures and I will be forever in your debt."

Hesitantly accepting their fate, Albert and Annie took up the tools and began their adventure into the LetterStorm.

2.3 Characters

Albert: Older brother of Annie, Albert is widely recognized as a geek. But don't let his square glasses fool you; his quick wit it more than a match for any challenge.

Annie: Younger sister of Albert, Annie is the most shy girl in her class. Her curiosity is her greatest strength, empowering her to overcome her shyness and be brave in the face of danger.

Charamons: Accidental creations of Professor P, these letter-shaped creatures now roam his lab and the surrounding hillsides. While most of them are docile, the six vowels are carnivorous and easily agitated. They can be defeated using the magic pencil and their pure form collected by the magic journal.

Professor P: Professor P is a highly experienced lab scientist devoted to the greater good. Unfortunately, his experiments sometimes get a little out of control…

Wordians: When a few charamons join together in the correct sequence, they can become a more powerful creature called a wordian. Wordians are rare but also difficult to defeat. They are most vulnerable to the pure forms of the letters composing them, when fired in the correct sequence.

2.4 Level Overview

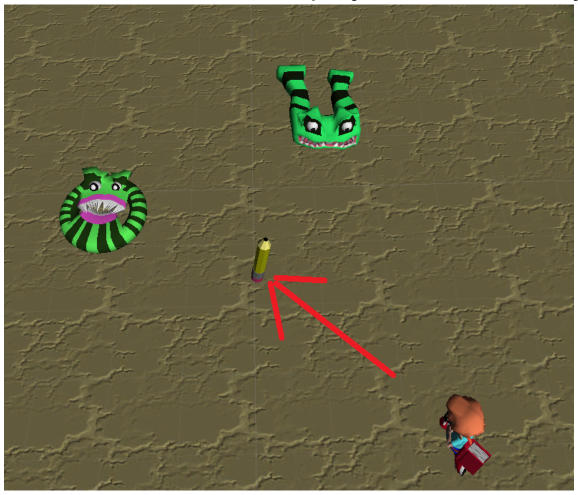
When the user starts the game, he selects a set of words he wants to learn how to spell, or a lesson. Each level is associated with one word in this lesson, with early levels in the game based on easy words and the latter levels on harder words.. The user's avatar starts out at the bottom of the screen facing upward and is automatically moved forward as the background scrolls down. Enemies are generated from the top and sides of the screen based off the characters in the level's word. All enemies attempt to damage the player, some via collision and others via weapons. The user can damage enemies using one or more of his weapons, with each enemy dropping a letter upon defeat. Once all the letters required for the word are collected, enemy generation stops, the boss arrives on-screen, and a word hint is displayed at the top of the screen. The user must defeat the boss by firing the collected letters at him in the right order to spell the level's word.

Since this game is a 2D scrolling shooter, gameplay environments do not play a significant role in the game itself. The game's story is set in and around a large lab building secluded within a lush, natural landscape. Earlier parts of the game will take place on a dirt road through a natural surrounding leading up to the lab. Later levels will take place within the lab building itself. It is difficult to make the environment progression more fine-grained than this, as the number of levels the user will play will be dependent upon which set of words he wants to learn, and he has the ability to change how many words are in any word set.

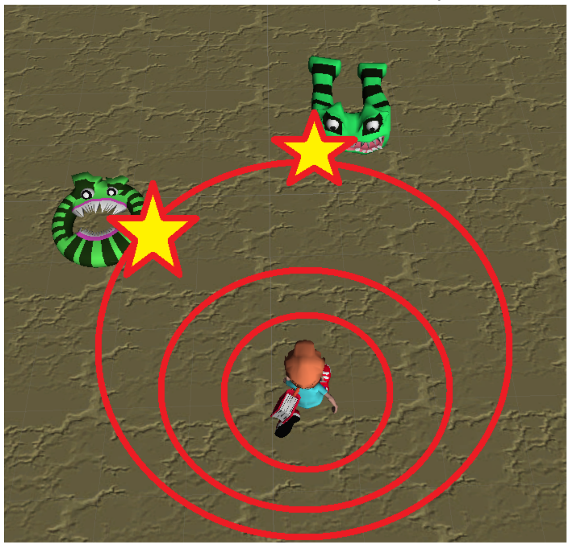
3: Gameplay - Paul/Nabil

3.1 Combat

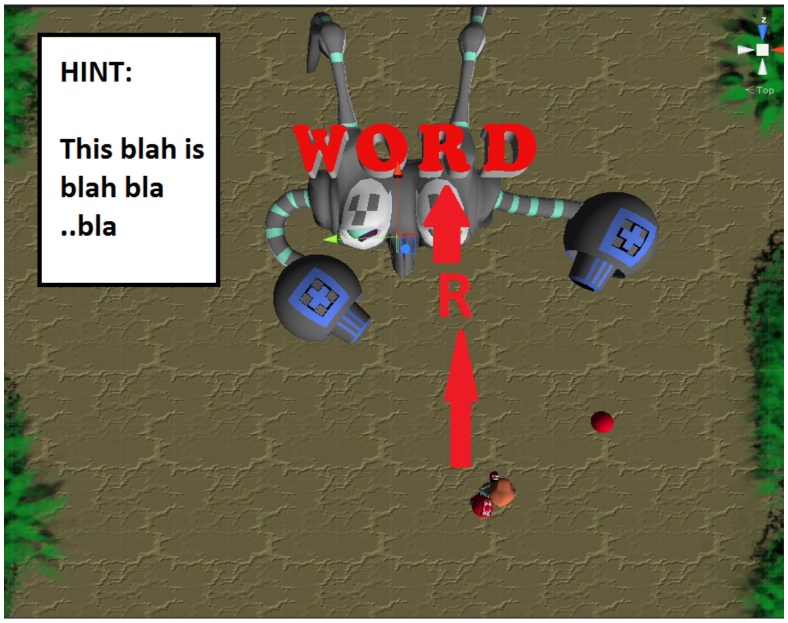
Projectile:
Albert/Annie can throw projectlie at a enemy in order to collect a letter drop.



Area damage:
Albert/Annie can use a different type of attack such as AOE.



Boss combat:
Albert/Annie will have to use the letters collected in the level in order to solve and spell out a word during the boss fight.



3.1.1 The Player Perspective

Each level pits the player against waves of letter monsters. The player is force-scrolled through the level, although within the player's field of vision he may move along a horizontal and vertical axis. The player may throw projectiles in order to defeat letter monsters. When the use-weapon key is pressed, a projectile is instantiated directly in front of the player's current position and travels in a straight line in the direction the player is currently facing until it either collides with an enemy or moves out of view, at which point it is destroyed.

3.1.2 The Enemy Perspective

There are two different types of letter monsters in Letter Storm -- Consonant Monsters and Vowel Monsters. If either Monster type moves below the bottom of the screen or a player's projectile collides with it, it is destroyed. If the player manages to defeat a Monster before it reaches the bottom of the screen, it will drop its associated letter for collection by the user.

Consonant Monsters are unintelligent enemies that move in variations of two basic patterns. Upon spawning, they may either choose to move along the vertical axis from the top of the screen to the bottom, or they may choose to move towards the coordinates the player was located in at the exact moment of its instantiation. Consonant Monsters do not have projectiles or attacks of any kind, and may only cause a player to lose a life if the enemy collides with him.

Vowel Monsters are the more threatening of the two basic enemies and pose a greater challenge to the player. Vowel Monsters may move sinusoidally, rather than follow a simple trajectory through the battlefield and then out of sight. In many cases, they move back and forth on screen firing their own projectile at the player. If a player collides with either a Vowel Monster or the monster's projectile, he will lose a life.

3.1.3 Bosses

At the end of each level, when the player has collected sufficient letters to spell a word that has been predetermined, a boss will appear -- recognizable as such by its massive size and imposing arsenal. The boss moves in a simple back-and-forth pattern while attacking with projectiles and melee weapons. The player must bombard the boss with projectiles until it goes into a vulnerable state, at which point it will not move or attack and appear dazed. At this time, the player may use the letters he has collected to fire special projectiles in order to spell the boss word correctly. After the boss is hit with the correct letter in sequence, it is no longer in a vulnerable state, once again moving and attacking the player.

3.2 Enemy Management

The Enemy Generator is an object that keeps track of how many enemies currently exist and decides whether or not to spawn enemies, and in what quantity to do so. The generator has an array of Spawn Points, and to create an enemy, will simply instantiate either a Vowel Monster or Consonant Monster as the child of a Spawn Point. Movement is not handled by the generator, but rather by the enemy itself. The frequency at which enemies spawn and the number of which may exist at any given time are determined by the difficulty setting that the player has chosen.

The generator also stores the word associated with the boss for the level, and keeps track of how many of the necessary letters the player has collected before ceasing its normal spawning pattern and instead instantiating the boss.

3.3 Collectibles

There are several types of collectibles that will appear as players defeat enemies. Enemies will drop the letters that they resemble, which players will be required to stock up on in order to face the boss. As mentioned previously, the boss will not spawn unless the necessary letters have been collected.

In addition to these, players will have the opportunity to collect powerups, which spawn infrequently and give the player a stronger means of attacking, or trophies in the form of classic books, which do not alter gameplay in any way but encourage with completionist tendencies to collect all of the “classics” and fill up their bookshelf.

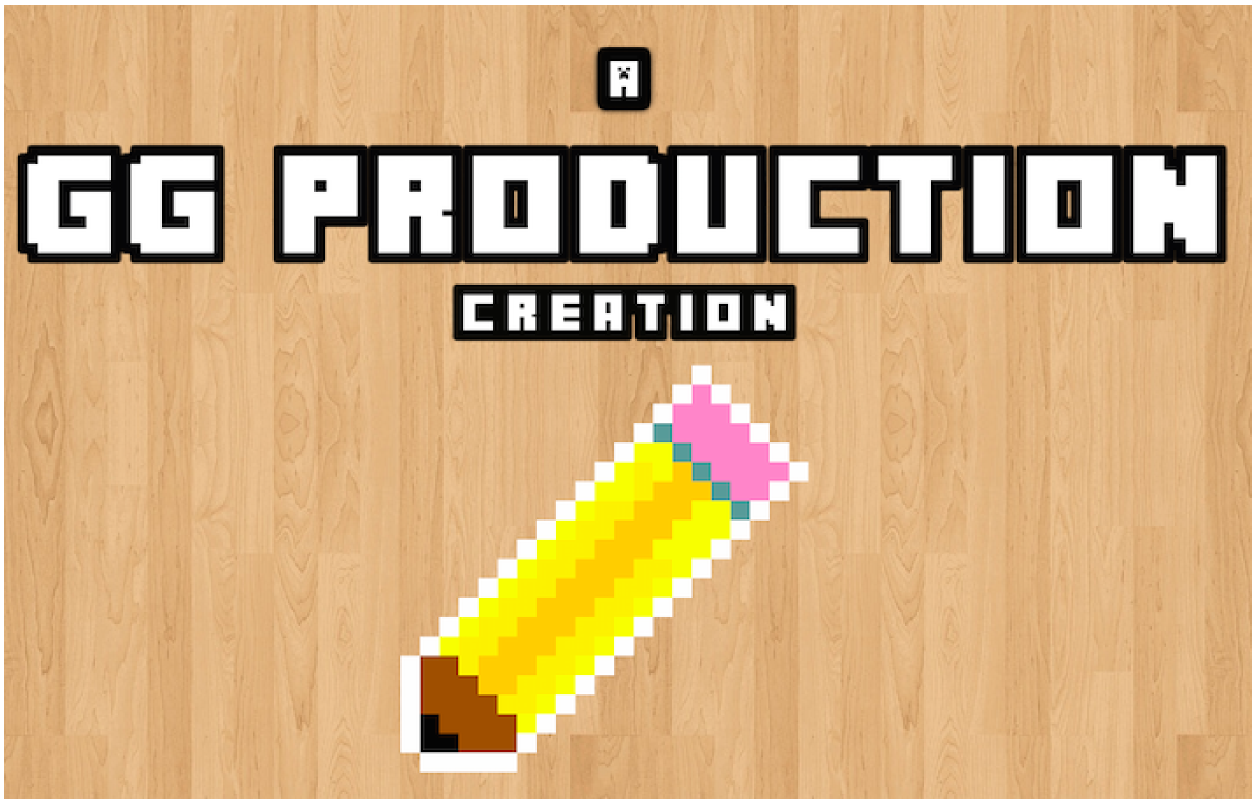
4: Menu Layout - James

4.1 Color Scheme

Color Schemes have not been implemented or explored as of yet. We are looking to explore this possibly next iteration.

4.2 Game Title Screen

Currently we have a Splash screen advertising our company name which leads into the “Main Menu / Title Screen”.



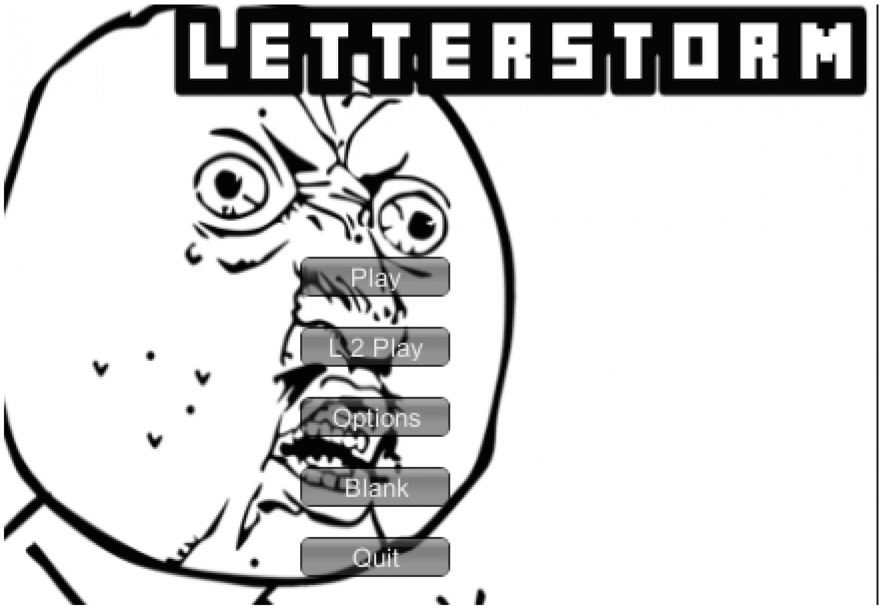
4.3 Legal info

The following license will be included with the game, as well as listed on the Credits screen.

4.4 Menu Screen

The Main Menu screen has 5 major buttons: “Play”, “Tutorial”, “Options”, “Credits”, and “Quit”. The “Play” button lets you select which word set you would like to learn and start the game from level 1. “Tutorial”, or “L 2 Play” in the prototype below, presents the user with instructions on how to play when clicked. “Options” is where players can customize the general game play settings such as colors and volume. “Credits”, or “Blank” below, links the user to Legal and Credit remarks. Finally, the “Quit” button allows the player to stop and exit the game.

The first iteration prototype of the Main Menu screen is pictured below. Please note that the prototype is not fully functional and will undergo significant layout and style changes before release.



4.5 Tutorial/Instructions Screen

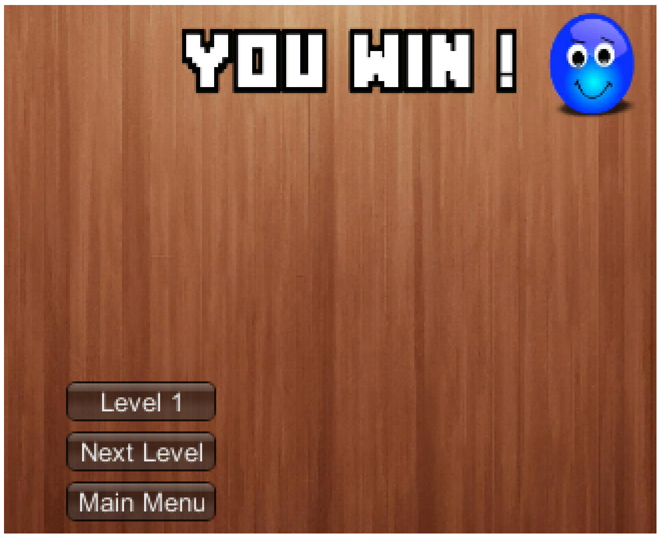
The tutorial screen has not been implemented, however it will be attached to the “Main Menu”.

4.6 Credits Screen

No credits screen has been added as of yet but is in the works for the second iteration. The thinking is to either have a credit button or splash screen appearing after the win screen.

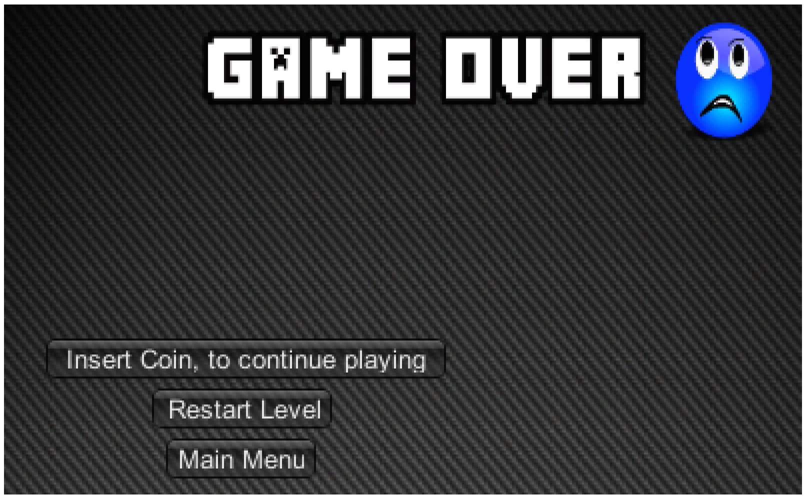
4.7 Game Win Screen

Upon successfully completing a level, the user will be greeted with the Game Win screen. It currently contains 3 buttons: ”Retry”, “Next Level”, and “Main Menu”. “Retry”, which will shown as “Level 1” below, allows the player to repeat the past level, resetting the score and lives. “Next Level” loads the next level. “Main Menu” returns the player back to the Main Menu, at which point the player could customize gameplay or quit the Game.



4.8 Game Over Screen

Upon losing all his lives, the player will see the Game Over screen. It contains 3 buttons: “Insert Coin”, “Retry”, and “Main Menu”. “Insert Coin” lets you pick up the game from where you died, letting you keep your old score. The conditions under which this functionality should be available are still being evaluated. The “Restart” and “Main Menu” buttons function identically to those on the Game Win screen.



4.9 High Score Layout

The possibility of score tracking has been considered in passing, but is considered a low-priority requirement. Should time permit, it might be motivational to the user experience to display a list of high scores on the “Game Over” screen or “You Win” screen, essentially showing your play statistics from the level in relation to other playthroughs.

4.10 New Game Screen

The New Game screen has not been added as of yet but is in the works for the second iteration. This is largely because the limited set of functionality developed during the first iteration did not justify the need for such a screen. The screen itself will be displayed when the user click the “Play” button on the Main Menu. It will allow the user to select which set of words he want to learn, the difficulty of the enemies, and which avatar he wishes to play with. Upon selection of these, he will be able to launch a new game starting at the first level from the screen.

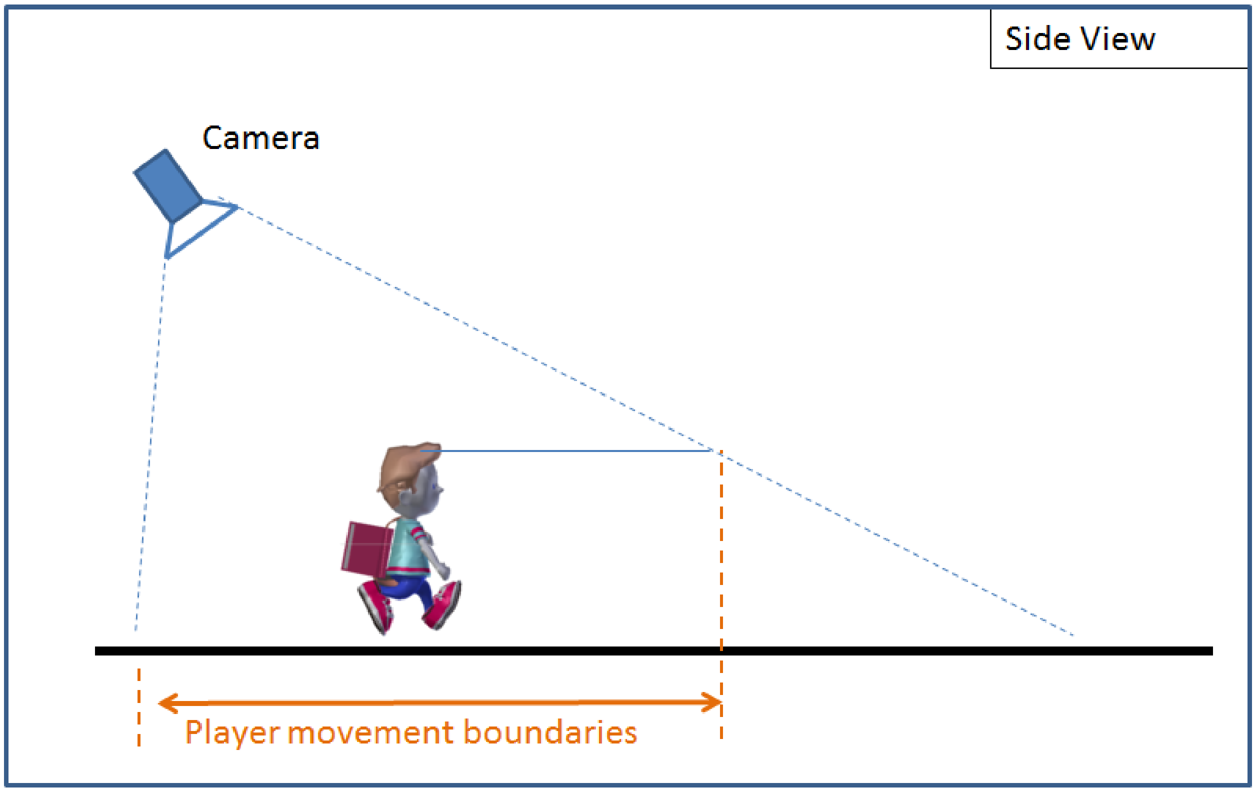
4.11 Lesson Management Screen

The Lesson Management Screen is allows the user to create new word sets or edit existing ones. The user may also specify the hints displayed for each word. It has not been decided whether this screen should be linked to from the Options screen or the Main Menu screen, though this is likely to be decided during the second iteration.

5: Game Layout - Nabil

5.1 Camera Setup

The camera remains stationary throughout the entire game. It is placed above the player, and it is tilted slightly to give a 3d effect.



5.2 Game Controls

5.2.1 Player Actions

The player has the following actions available to him:

- Move
  - Forward, into the scene
  - Backward, toward the camera
  - Strafe left with slight angle
  - Strafe right with slight angle
- Attack
  - Use equipped weapon
  - Fire letter at boss

        - Select letter to fire
        - Fire letter
    - Change weapon
        - Change to specific weapon
        - Cycle through weapons

5.2.2 Control Mapping

Upon release, multiple control schemes will be supported. The two most likely to be included are detailed below.

Keyboard-only

- Move
    - W or up arrow: forward
    - S or down arrow: backward
    - A or left arrow: left
    - D or right arrow: right
- Attack
    - Space: use weapon
    - Fire letter at boss
        - L + any character key: select letter to fire
        - Space: fire letter
    - Change weapon
        - number key: change to specific weapon
        - tab: cycle through weapons

Mouse

- Move
    - Main character will follow the mouse cursor
- Attack
    - Left-click: use weapon
    - Fire letter at boss
        - Click the letter in the inventory: select letter to fire
        - Left-click: fire letter
    - Change weapon
        - scroll-wheel: cycle through weapons

5.3 Game Mode(s)

The game will have one playable mode with variable levels of difficulty. There are two factors that affect game difficulty: enemy difficulty and word length. When the user starts a new game, he will be able to select the enemy difficulty (easy, normal, hard) and the lesson to practice. The enemy difficulty affects such properties as the enemy health, enemy speed, and enemy density. It will optionally affect how many lives the player starts out with, and how often he is presented with power-ups. The lesson selected will indirectly affect how many enemies must be defeated before reaching the boss, how many letters must be used to defeat the boss, and how many levels must be completed to win the game.

5.4 Player Count

The game will feature 2 protagonists: Albert and Annie. Only one protagonist can be played at a time; multiplayer of any variety is not supported. The player will be able to select his protagonist from the New Game screen. There will be an optional hidden character: Patrick the Platypus . This character will only be available after achieving a specific high score or completing a specific amount of lessons successfully.

5.5 Hours of Gameplay

Letter storm will feature a finite set of levels that can be repeated multiple times. The levels will get harder as the player progresses. While the amount of time spent in the game will be directly related to how many words are in the lesson the user is learning, their complexity, and the chosen enemy difficulty, it is not expected that a level should take more than 10 minutes to complete.

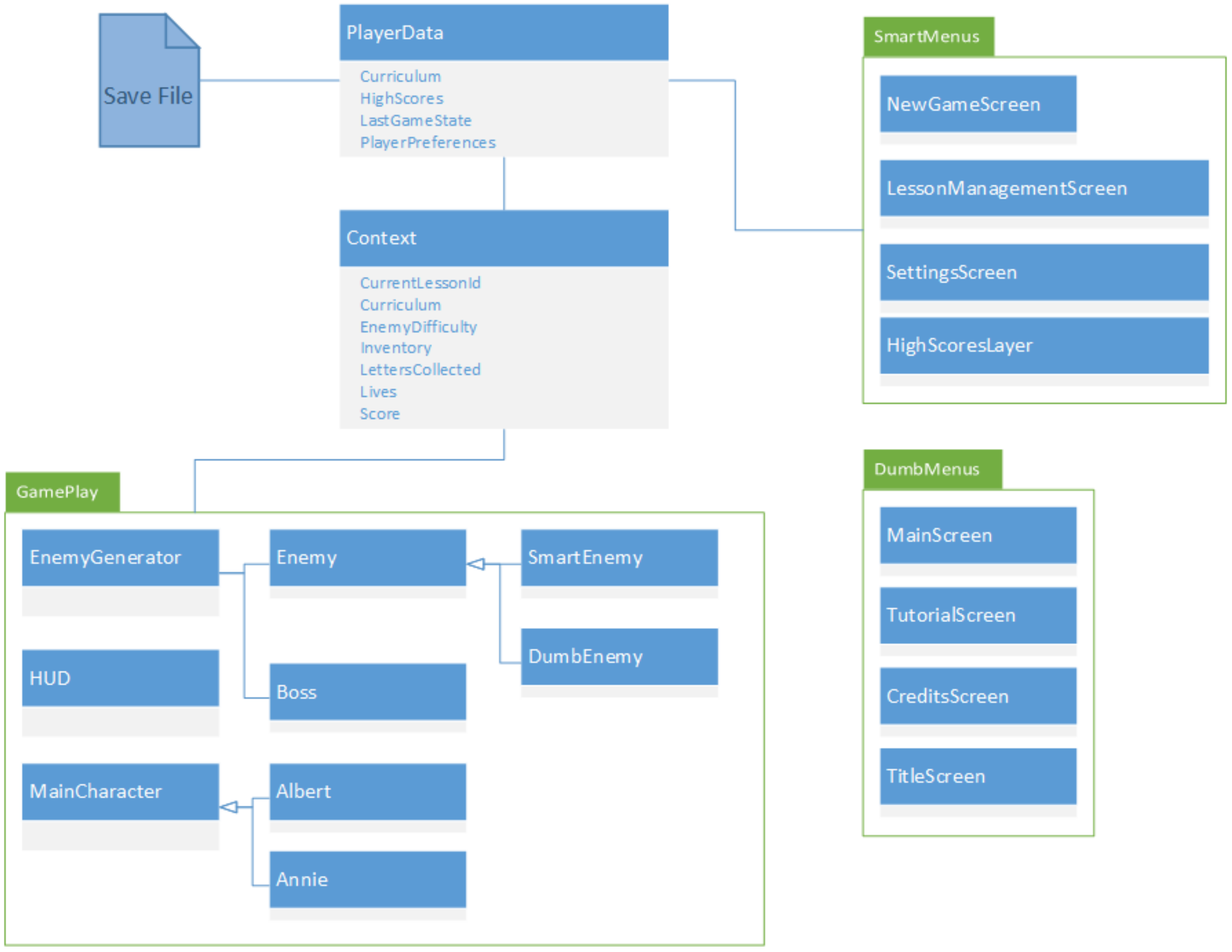
6: Code Design

6.1 Design Model Overview

This code base for Letter Storm is being developed with the MVC architecture in mind. Due to the variation of objects present in a complete game and their varying dependence upon data, though, some objects will simply follow either a VM pattern (no controller). Please see the sections below for more details.

6.2 Data Flow

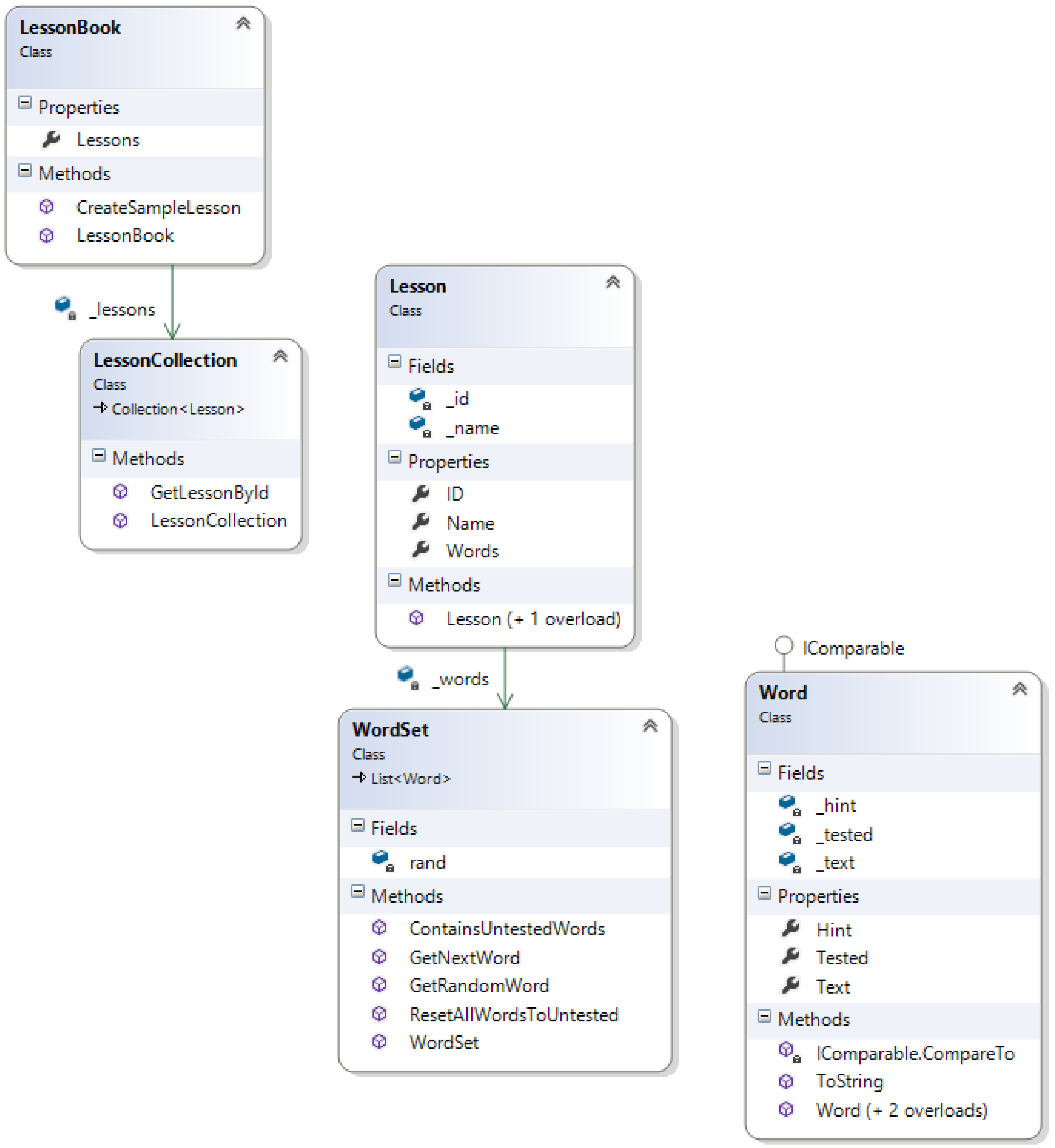
In general, persistent data is stored in a save file. It is loaded into the PlayerData object, which acts as an interface to the file. Upon starting a new game, relevant data present in the PlayerData object will get copied over into the game's persistent Context object. The Context is created when a new game is started and is destroyed when the player has to return to the Main Menu. The EnemyGenerator takes words from the user-selected lesson stored in Context to generate a level's enemies and boss. The Context is also used to keep track of the game's current state, including such things as the current score, how many and which letters have been collected, how many lives the player has, etc. Outside of gameplay, some menus interface directly with the PlayerData object load and save data such as the user's lessons, high scores, and preferences.



6.3 Data Constructs

6.3.1 Lessons

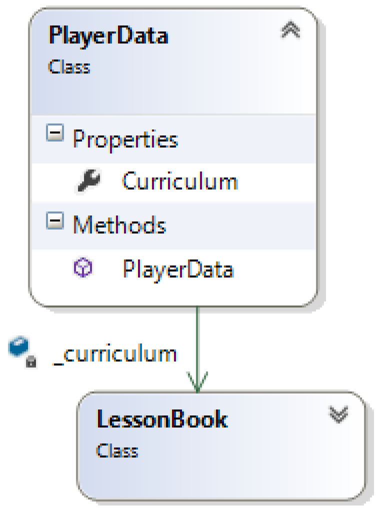
Users can create sets of words to be tested on. These words and lessons are stored and accessed using the following class constructs:



6.3.2 Player Data

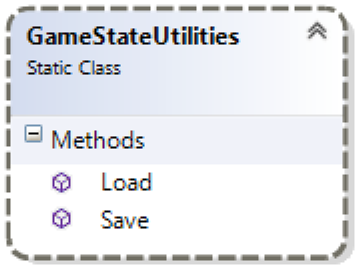
Lessons, high scores, and the game state need to be grouped and stored in a file to save across play session. The PlayerData class is used to group this data for saving and loading.

Please note: The diagram below is based off progress during iteration 1. It does not represent the final structure of the class, as additional functionality has yet to be added.



6.3.2.1 Game State Utilities

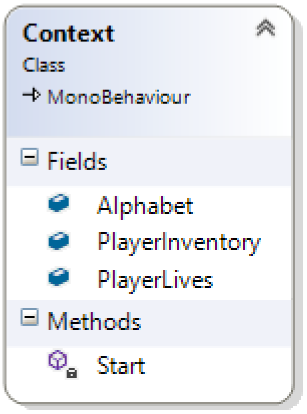
The static GameStateUtilities class is used to load the PlayerData from file and/or save it back.



6.3.3 Context

The game Context is used to keep track of the game state during gameplay. It is also used to store data that needs to be accessed by multiple independent entities, such as the HUD and the EnemyGenerator. The Context is created when the user loads his first level and is destroyed when the user returns to the Main Menu. Some if its properties, such as Curriculum, as always populated from the PlayerData object. Its other properties, like the EnemyDifficulty or Lives, may be populated either dynamically based on user actions or from the PlayerData object when loading a saved game. If the user chooses to save his game, select properties of the Context will be copied back to the PlayerData object before it is saved to file.

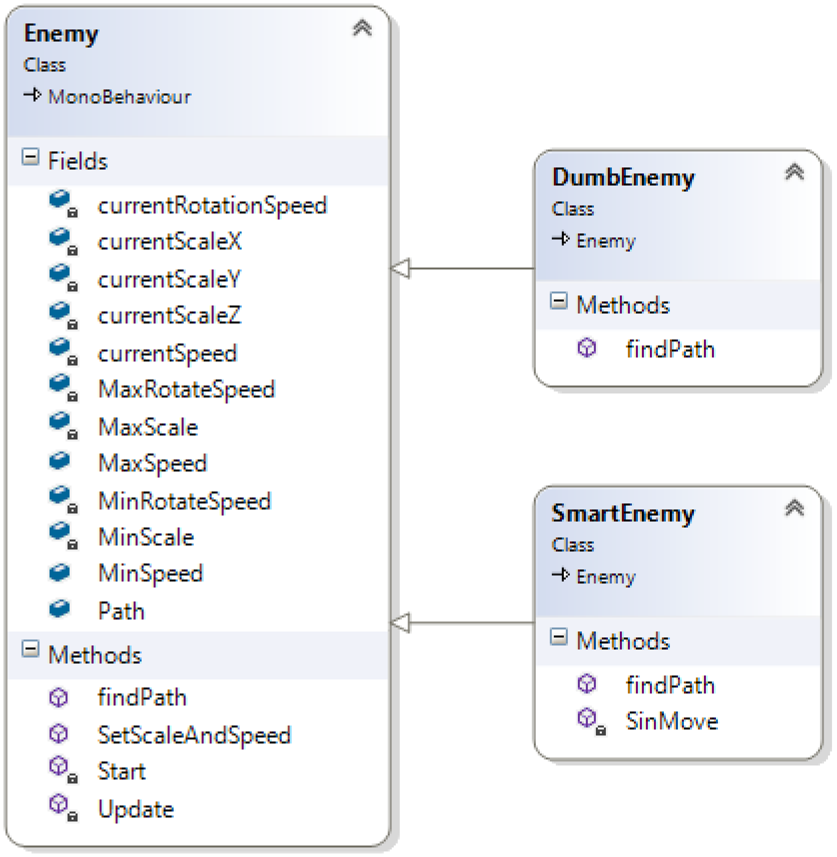
Please Note:The diagram below is based off progress during iteration 1. It does not represent the final structure of the class, as additional functionality has yet to be added and/or refactored.



6.4 Game Objects

There are three main types of game objects: Enemies, MainCharacter, and HUD. Each of the objects have access to the Context and share data using it. For example, if the MainCharacter is hit by an Enemy, the MainCharacter will update the Lives count in the Context, which the HUD will then retrieve and display.

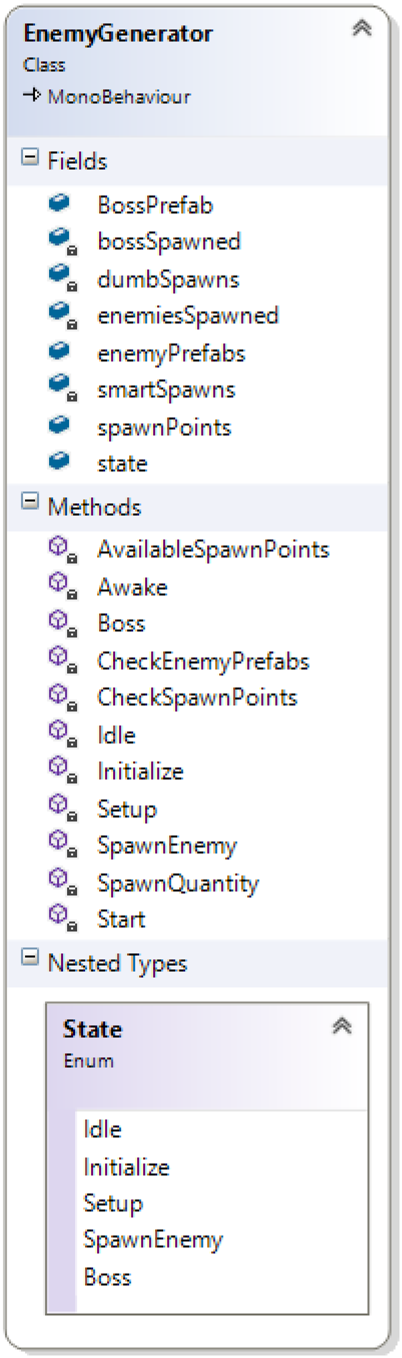
The game objects themselves do not have strict separation between View and Controller logic in the traditional sense, though. There is very little View code used, since what is visible is actually 3D models generated using Blender. The Controller code is a script attached to these models that instructs them on what actions to take based on the Unity event system.



6.4.1 EnemyGenerator

The EnemyGenerator is a special game object that controls what happens within a level. It interfaces with the Context to determine what word the user should be tested on. Using letters from this word, it randomly generates enemies associated with some of these letters at random locations off-screen. It also monitors what letters the user has collected, as listed in the Context, to determine when to display the level's boss.

Please Note: As with most class diagrams, this one is not final and is subject to change.



6.5 Menus

Unlike game objects, menus must be built completely via code. The creation of menu objects and the monitoring of events associated with them is so tightly coupled, though, that it is significantly difficult to separate view and controller code for them. Thus, all existing data-dependent, or smart, menus use a model-view architecture (vs MVC), with static menus existing fairly independently via a view architecture.

Smart menus include the New Game screen, the Lesson Management screen, the Settings screen, and the High Scores layout. All other menus are static. Smart menus have direct access to the PlayerData object, as there is no Context object loaded for them to utilize. Most smart menus (excluding the New Game screen and High Scores layout) can save their changes back to the PlayerData object so they can be loaded later.