Game Design Document

1. Game Overview

The game is a fast paced fantasy themed 2D side scrolling combat platformer. The levels will take place in a number of fantasy themed environments. In this game the player controls a single character and must reach the end of each level while defeating a number of various enemies, avoiding traps and other dangerous obstacles (spikes, blades, etc.) and collecting items that add to the gameplay in a meaning full way, such as enhancing the players abilities, returning lost health, allowing the player to create a checkpoint at almost any location or allowing the player to overcome an obstacle or puzzle. The player will have an initial number of lives which will be decreased upon death. There will be a health bar to display the player’s health. The player can regain health by collecting health orbs, reaching a checkpoint or defeating enough enemies using the environment. The player will be confined to melee attacks for the majority of the game though certain collectibles will allow the player to perform ranged attacks for a short period of time.

**Influences**

This game is influenced by a number a existing side scrolling platformers such as "Vvvvvv" and "They Bleed Pixels". Both of these games influenced the style and usage of traps, such as blades and spikes, in my game, while also influencing the pace of the gameplay. However though these games influenced the pace and gameplay of my game, they themselves can be extremely difficult at times and my test the patience of the average gamer. As such my game may be similar but will not feature difficulty spikes of their magnitude. The combat in my game is heavily influenced by that of "They Bleed Pixels" in that the player is encouraged to use their surrounding environment to defeat enemies. This includes large drops, and nearby trap or obstacles and other enemies themselves. (explosive enemies, etc.)

2. Game Mechanics

2.1 Space

As a 2D platformer the games space will consist of a number of continuous levels progressing from left to right. The space will have at least three boundaries at any time, left and right to prevent the player form leaving the level, and the floor. There will be a ceiling at times to force the player into more narrow spaces. This will make avoiding damage more difficult in these areas. As a 2D game each level has 2 dimensions. Throughout the game the player may be given the option to enter sub-spaces such as caves or buildings of various sizes. These sub-spaces will essentially be smaller levels contained within the larger level and are completely optional. As such the player will be rewarded for completing any of these sub-spaces. They are connected to the larger level by entrances such as doorways. To enter a sub level the player must approach the entrance and will then be prompted to push a button to enter.

2.2 Objects

Objects contained within the game world are:

**Enemies**

**Attributes:**

An enemy in the game has the following attributes; health, position, state (idle, attacking, fleeing, etc.), strength, alive, armed, ttl.

**Attribute States:**

Health: A value between 0 - 100 indicating the enemies current status.

Position: A 2D vector containing the enemies current position.

State: Set to idle, attacking, fleeing, etc, to alter the enemies behaviour.

Strength: An integer value to determine how strong the enemies attacks are.

Alive: A boolean value set to true if the enemy is alive, false if it is not.

Armed: A boolean value to determine if an explosive enemy has been triggered.

Ttl: An integer for explosive enemies. Indicates time until detonation.



**Traps**

**Attributes:**

Traps will have the following attributes; position, active, alive.

**Attribute States:**

Position: A 2D vector containing the traps position.

Active: Set to true or false. Decides if a trap is active or not.

Alive: Set to true or false. Used to check if the trap is destroyed.

**Ability/Health Altering Collectibles (speed, attack strength, etc)**

**Attributes:**

Collectibles have the following attributes; position, effectiveness.

**Attribute States:**

Position: A 2D vector containing the traps position.

Effectiveness: An integer which affects the potency of this power up.

The player will not know how effective a pick up is until they have taken it. They will then be informed visually of its affect.

**Player**

**Attributes:**

The player will have the following attributes; position, health, strength, max speed, alive, jumping, direction, velocity, acceleration.

**Attribute States:**

Position: A 2D vector containing the player’s position.

Health: A value between 0 - 100 indicating the players current status.

Strength: An integer value to determine how strong the player’s attacks are.

Max Speed: An integer which determines the player’s max movement speed.

Alive: A boolean set to true if the player is alive, false if they are not.

Jumping: A boolean set to true when the player jumps, false when they land.

Direction: A 2d vector to store the player current direction.

Velocity: A 2d vector to store the player current velocity.

Acceleration: A 2d vector to store the player’s acceleration speed.

The player will need their current health value displayed on screen, in the form of a health bar, as there will be no visual way to tell its current state. There will also need to be a visual representation of the player’s current attack strength. The player could be scaled up or change colour for example.

**Power up Chests**

**Attributes:**

Chests have the following attributes; position, tier

**Attribute States:**

Position: A 2d vector to store the chests position.

Tier: An integer between ~1-5. Affects the potency of the chest’s contents.

**Doors**

**Attributes:**

Doors have the following attributes, position, leadsTo.

**Attribute States:**

position: A 2d vector to store the doors position.

LeadsTo: Used to load the sub level associated with this door.

The player will need to be informed when they approach a door. This will be done through a button prompt to enter the sub level.

2.3 Actions

**Actions:**

Move, jump, punch, kick, open, enter, avoid, defeat, climb.

The player can move around the game world by moving left, right, jumping, climbing ladders, dropping off ledges, etc. The player jumps to avoid enemy attacks, traps, solve puzzles jump over holes, etc. The player may punch enemies to deal damage or kick them to propel them away. Enemies may be kicked into traps, of ledges or into each other, stunning them briefly. The player may open chests found through the game. These chests drop a random power up for the player. The player may enter some sub levels such as buildings or caves. The player will need to avoid traps and enemy attacks throughout the game. The player will defeat enemies. The player can climb ladders to reach higher areas in some levels.

The player may be required to combine two or more of these actions to progress. For example the player may need to move(run) then jump to surpass some obstacles.

**Operative action Result**

Open chest. A random power up is spawned.

Kick. The kicked object is propelled away from the player.

Enter. The player enters a building or cave.

Jump. The player is propelled into the air.

Move. The player character moves.

2.4 Rules

The rules of the game will be consistent throughout each level though may vary slightly between levels. However some rules will never change, such as, players will always be killed immediately by active traps, enemies will always attack the player in sight, the player may only have one active power up at any one time, enemies may always be defeated, gravity will always act downwards and the player may only perform two consecutive jumps before hitting the ground. An example of a rule that may change could be, in an icy level the ground may be more slippery, making avoiding traps more difficult.

The problem the game asks the player to solve may vary per level. Some may require the player to time their movements to avoid obstacles, other may be more combat orientated and may require the player to defeat enemies to progress.

Though some of these problems will be more difficult than others, the player will always feel in control. They will not be tasked with solving a problem that they believe cannot be solved.

The player will be encouraged to take risks throughout the game in order to achieve a higher score. If the player reaches a sufficiently high score they will be rewarded with new game modes, difficulty levels or cheats such as invincibility.

The player completes a level by reaching the end point, which will be clearly marked.

2.5 Skills

The skills required for this game will primarily be mental skills and timing skills, though quick reflexes will also be required. Mental skills will be required to overcome some puzzles and obstacles while good timing will be needed for combat and avoiding traps. The time a player takes to complete a level will factor into their final score. The physical skills required will simply consist of the player interacting with the target device in order to play the game. (pushing buttons, using the accelerometer, etc.).

2.6 Chance

One way in which chance plays a part in this game is in the form of chests found throughout levels. These chests will drop a random power up for the player when they are opened. These will be the most random part of the experience.

However, the player will also be able to take chances such as declining to place checkpoints during a level. This creates a risk/reward mechanic because the player will be rewarded if they place a low number of checkpoints during a level. However, if the player dies without having placed any checkpoints they will be forced to play the entire level again. Because it is the players choice to forgo some or all checkpoints, the magnitude of the risk taken is up to the player, and of course a higher skill level is required to complete a level without any checkpoints.

3. User Stories

* **Menu Screen**

As a user I want a menu screen so that I can easily access all available options.

**Conditions of satisfaction**

* I can continue my most recent games easily.
* I can access a list of unlocked levels.
* I can quit the game.
* I can start a new game.



If the player chooses continue the latest saved game will be loaded.

New game will start a new game and overwrite the current saved game. The player will be asked to confirm this action.



Replay level will display a list of currently unlocked levels. The user then chooses the desired level from this list.

Quit will exit the application.

* **Continue Game**

As a user I want to easily resume my latest saved game.

**Conditions of satisfaction**

* The correct level is loaded.
* My position, stats and progress have been loaded correctly.
* Defeated anemies and collected items do not reappear.
* **Load/Save Game**

As a Developer I want game data to be saved to/loaded from a save file.

**Conditions of satisfaction**

* Game data can be saved to a file.
* Game data can be loaded from a save file.

Saved data will include enemy and item positions, player score, player health, player position, player progress.

* **Generate level.**

As a developer I want to generate all level geometry from an image and position all enemies and items based on the contents of an xml file.

**Conditions of satisfaction**

* All level geometry is created based on the pixel data of a corresponding bitmap image.
* All enemies are created, positioned and initialised, based on the contents of a corresponding save file.
* All items in the level are created and positioned based on the contents of a corresponding save file.
* **Replay Levels**

As a user I want the ability to easily replay levels.

**Conditions of satisfaction**

* I am shown a replay level option.
* I am given a list of unlocked levels.
* I am shown performance data for each completed level.



From the main menu the user selects the replay level option.

Main menu will return to the main menu screen. Play selected level will play the currently selected level. The player stats from the selected level are displayed on the left. Unlocked levels are shown on the right.

* **Saved Games**

As a user I want my game progress and stats saved so that I can view them later and continue a game from where I left off.

**Conditions of satisfaction**

* My progress data is saved.
* I can view my stats from the main menu.
* I can easily continue my latest saved game.

Stats can be view per level from the reply level screen covered earlier.

Games can easily be resumed by selecting the continue option from the main menu screen as previously covered.

* **Pause Menu**

As a user I want a pause menu do that I can leave/save the game at any time.

**Conditions of satisfaction**

* I can easily pause the game, and am shown a menu.
* I can resume the game.
* I can quit the game.
* I can save the game.

Resume simply resumes the game.

Save will overwrite the latest manual save.

Quit will return to the main menu. The player will be asked to confirm this action.

* **Level Complete Screen**

As a user I want to be shown my performance at the end of a level and be given the option to retry the level.

**Conditions of satisfaction**

* I am shown a level complete screen
* I am shown my performance information for the finished level
* I am given the option to retry the level.



Any option will automatically save these stats if they are higher than the player current high scores.

Continue will proceed to the next level.

Retry will restart the current level.

Quit will exit to the main menu. The player will be asked to confirm this action.

* **Grid Based Levels**

As a developer I want grid based levels so that I can organise the layout of the levels faster and easily access info about a specific grid square and its contents.

**Conditions of satisfaction**

* The game world is organised into an n\*n grid.
* Each grid square holds info about what that square contains.
* I can easily access info contained within each grid square.
* **Climbing**

As a developer I want multiple methods of climbing so that there is more complexity to the game.

**Conditions of satisfaction**

* The user can jump or double jump onto high platforms.
* The user can climb ladders to reach high platforms.
* The user can climb and jump from ladders to reach distant high platforms.
* The user can grab walls in order to climb vertical surfaces.
* Single jump to lower platform,
* Double jump to higher platform.



The player can grab and jump from vertical walls to overcome some obstacles.

* **Parallax Scrolling**

As a developer I want parallax scrolling of the background in the game so that it adds some depth to the otherwise flat scene.

**Conditions of satisfaction**

* The background should scroll slower than the foreground.
* Multiple scrolling layers.



As the player moves objects in the foreground (box) scroll faster than objects in the background (Tree). Some levels may have more scrolling layers; the “deeper” a layer is the slower it will scroll.

* **Difficulty Level**

As a user I want a choice of difficulty level so that the game is not too difficult.

**Conditions of satisfaction**

* When I am about to start a level I am asked to choose a difficulty level.
* There is a sufficient difference between difficulty levels.



Just before the player begins a level they will be asked to choose a difficulty level.

If the player completes a level and chooses the continue option on the level complete screen, they will not be asked this. The next level will use the active difficulty level automatically. This will maintain a steady difficulty which will help maintain a good flow.

* **Traps**

As a player I want to clearly see any dangers in the game world.

**Conditions of satisfaction**

* Enemies and traps are clearly visible.
* The magnitude of danger is clear.
* The range in which traps can harm me is clear.



The player must avoid the traps A and B by following the path laid out by the green arrows (Arrows will not be present in the game). The danger is clear, the traps range is clear.

* **Pick ups**

As a user I want any pickups to be clearly visible and to stand out.

**Conditions of satisfaction**

* Pickups are clearly visible.
* Their effect is clearly displayed when they are picked up.



When the player picks up a pickup it will be clear that they have done so. The effect of a pickup will be clear by the image used to display it (A fist for increased damage, a running shoe for increased speed, etc.).



The player will need speed boosts to clear some obstacles in the game.

* **Power up effects**

As a user I want to know the effect of a power up I pick up so that I can use it effectively.

**Conditions of satisfaction**

* The power ups effect is displayed clearly.



Example of a speed boost.



Example of a damage boost.

* **Controls**

As a player I want a minimalistic control scheme so that the buttons do not clutter the screen.

**Conditions of satisfaction**

* Clear controls.
* Buttons take up the least amount of space possible while still being effective.
* Simple control scheme.



The two buttons on the left move the player’s character.

* Performs a heavy kick attack that propels objects and enemies.
* Jumps.
* Punches.

If the player moves against a wall while in the air they will grab it. This grab feature is used as described earlier.

The player will always be in the centre of the screen so as to avoid being covered by these buttons.

* **Movement**

As a user I want to be able to move around the game world.

**Conditions of satisfaction**

* Clear controls.
* Clear level boundaries.
* Responsive controls.

The player may move around the level using the controls as described earlier. Gravity will always move the player downward if they are in the air.

This shows some of the ways in ways in which the player may defeat enemies.

* The player can avoid enemy attacks. This requires good timing.
* The player may use their environment to defeat enemies. By using traps for example.
* **Health state**

As a user I want a clear indication of my current health state.

**Conditions of satisfaction**

* Clearly displayed health state
* I am informed when my health state changes.
* There is a clear method of increasing my health.



The players health will be displayed on screen and can be increased by picking up health orbs throughout the levels.

* **Object States**

As a user I want a clear indication of object/enemy states.

**Conditions of satisfaction**

* The state of an object or enemy is visible clear.

It will be clear to the user whether or not enemies are alive or if traps are activated. Blades will not spin, enemies will be dead or de spawned, etc.

* Enemy is alive
* Enemy is dead
* Blade spins while active.
* Blade does not spin while inactive.
* **Level Ending**

As a user I want to know when I have reached the end of the current level.

**Conditions of satisfaction**

* A visual clue showing the end of the level.



Each level will have a clue indicating the end has been reached.

* A door shows the exit in the example above.