Sneaky Knight Game Design Document

Table Of Contents

Section 1 Game Overview

1.1 Game Title

Sneaky Knight

1.2 Game Genre

Our game is a primary platformer with both stealth and rpg elements. The setting is in a medieval fantasy world. The visual style would be under an 8 bit category with most of our sprites being based upon a 64x64 square with simple colors.

1.3 Game Perspective

The player and as a result the world around them is viewed from a 3rd person perspective in a 2d word allowing us to see everything that surrounds the player. Since the levels are large only the surroundings of the player in a 10 block area left and right as well as 8 blocks of vertical view.

1.5 Target Audience

The target audience would mostly be those between the age of 16 to 35 looking to relax playing a low challenge game and enjoy both the visual playstyle as well as the interesting gameplay mechanics.

1.7 Goal

The goal of the game is to progress through each level towards the boss in order to beat the game. A side objective is to increase the score by killing enemies either through stealth or combat but the game is in theory beatable with a score of zero if the player avoids every enemy. The player is also discouraged from combat in that health transfers over between battles while stealth killing an enemy prevents player health from dropping while eliminating an enemy.

Section 2 Game Background & Flow

2.1 Background Story

Main boss is expanding their kingdom through force to take over a neighboring kingdom as the previous ruler has passed away. PC's hometown is the most recent of conquests in this expansion.

2.2 Character Starting Story

Main boss ransacks town and takes PC's parents. PC meets mentor figure in a forested area. Mentor tells PC about the problems of the kingdom being a result of the Main Boss' rule and where he can be found. PC starts out by defending their hometown and progressing through cities until they reach the Boss' castle defeating them and saving their family.

Section 3 Game Play

3.1 Objectives

The primary objective of gameplay is to progress through the levels by reaching the right side of the level. This is perfectly attainable simply by running past enemies without them seeing you, but this does not result in any points. Thus, the secondary objective is to get as many points as possible by killing enemies. The easiest way to do this is sneak up behind an enemy and press space, but if the enemy turns around and sees the player then this will trigger combat. Finally, a third minor objective is to utilize double jumping and dashing to explore the map and collect hidden items.

3.2 Game Logic

3.2.1 Combat and Enemy Logic

Although our game is a platformer most of the time; if the player is "seen" by an enemy, then it will trigger a combat event and move to a new room. An enemy will detect the player's presence if the enemy is facing towards the player and the player is within the view_dist of the enemy. When the player is within this range, the enemy's sprite will update to reflect that it can see the player. If this continues for wait_time, then the enemy will bring the player into combat. In this pokemon-style combat room, the player and the enemy take turns attacking each other. The player has the option to

attack or flee, while the enemy has the option to attack or heal. Fleeing is not a guaranteed action, however, as explained further in section 4.2.1. Attacking and healing both have a random element to them, adding a small uncertainty to the amount of damage dealt or the HP healed.

3.2.2 View Logic

To keep the player sprite as the center of focus for the user and reduce the amount of visual stimulation we used a small view of the level to show the players surroundings. This was done with gamemakers in-built viewport code with the object to follow set as the player. In addition all relative position based text calls upon getting the viewports location and size before adding an offset to display in the proper location.

3.2.3 Performance Enhancement

For our larger levels with many enemies there is a noticeable increase in computational requirements as every enemy has timers, is searching for the player, and going through their idle animations. One of the first performance enhancements we did was short circuiting all of our detection logic by first checking some of the default booleans such as is the player hidden or is the enemy able to spot before going into the functions that calculate player location and if they are in line of sight. This gave some slight performance improvement but the sheer amount of enemies and size of the level still gave lag on smaller computers like laptops.

The largest performance enhancement was inspired by a video on how to do combat levels in the game. The key snippet of code was an instance_deactivate_all that stopped all objects from going over their step and draw functions. As a side effect this prevents all objects besides the one that called it from moving. The trick was making sure that enemies inside the view were active and looking at the gamemaker docs revealed another function instance_activate_region which used in conjunction with our view parameters allowed for everything within the view of the player to be activated. However since our controllers are static in their placement special calls had to be made to re-activate those objects specifically for our rendering script to work. After adding it to the player which we knew would always be in the view and as such not vulnerable to being deactivated we realized that the deactivating/activating took a large amount of time. However since the player would not reach the end of the view even in multiple calls of the step function a delay was added so we only render every 10 steps which reduced the overhead and gave us the same performance from the small tutorial level in the much longer city level.

3.2.4 Environment Logic

To handle interactions with the environment we decided to simplify how interactions are made. First the player is the only object with movement capabilities allowing all enemies and collectables to be in a static location not needing to handle any collision logic. Next all of the visual elements are sprites part of a tileset. The player handles most of its own interactions, being responsible for not falling through floors, walking through walls, applying gravity, and hiding behind objects.

3.4 Game Resources (Score Strategy)

To increase the score a player must kill an enemy. The player has two options for killing an enemy via either the stealth mechanic or through combat. Due to the nature of combat and a non-refillable health bar the best strategy is to avoid all enemies through the stealth mechanic and get close enough to kill them. While completely avoiding the enemy does allow for progression to the next level, it does not increase the score. This encourages the player to kill every enemy they encounter but avoid combat in order to preserve their HP and lives.

3.5 Game Progression

By reaching the far end of the level the player is transported to the next level with reset health and lives. The player gets reset to the beginning of the level after losing in combat or falling off the world, getting a reset to the health bar and losing a life. After losing a life the level is not reset so any enemies killed will remain killed pseudo-saving the players progression as well as making earlier previously completed parts of the level easier to re-complete.

Section 4 Game Elements

4.1 Environment

4.1.1 Environment Design

Every aspect of the environment was designed with the intent to blend gameplay with passive storytelling.

Tutorial: Here, the intent was to both showcase mechanics in a compact space and to show the player character entering a city from the wilderness beyond. To this end, as the player progresses from left to right, the scenery goes from grass & trees to bricks & a building, with more enemies appearing as they get closer to

the city. At the same time, each of the various mechanics was given its own purposeful showcase.

City: Here, the intent was to present a hostile environment that told a story of a mismanaged city. To this end, significantly more enemies were included, encouraging using other mechanics to go around them, while showing things like roads in disrepair and collapsed houses, tending to get worse as the player progresses through the level, and past the city park transitioning to a darker stone associated with the castle, creating a level of ominous atmosphere.

Castle: Here, the intent was to provide a more platforming-centric challenge, rather than the enemy-heavy design of the city, and in so doing allow the empty and precarious dark castle environment to build up a sense of apprehension and finality as the player approaches the final boss and the end of the game.

4.1.2 Environment Features

- **Multi-layered background:** Multiple layers of tilesets were used to design more complex and interesting environments from a limited pool of sprites.
- **Integration of hiding places:** The sprites chosen for the hiding mechanic, and their placement, was done with the intent to facilitate gameplay while not being out of place in the environment itself.
- **Platforming collectibles:** Though true collectibles were not implemented, there is still an additional reward for exploration and platforming in the form of health sources to recover from dangerous combat encounters.

4.1.3 Environment Challenges

Certain aspects of the environment were used as an active challenge to the player.

Avoiding enemies: In many places, the best way to handle the large number of enemies was to find the appropriate path of jumps to go around them.

Difficult jumps: In certain locations, the platforms are placed such that they are not trivial to reach, and failure means landing in a dangerous situation. In the city, this is accomplished by placing enemies below the jump, while in the castle, this is accomplished by simply allowing the player to fall to their death.

4.2 Characters and Objects

4.2.1 Playable Characters

The player character has a basic knight sprite with some basic parameters as seen below.

1. **Health Points:** Starts out with 100 health that carries over between battles.

2. **Speed:** Has a basic movement speed of 4 pixels per step which double to 8 with sprint
3. **Jump Speed:** The player jumps with an initial velocity of 12 pixels decreased by gravity
4. **Gravity Factor:** The player is affected by gravity at ½ a pixel per step cumulative
5. **Attack Damage:** The player has a basic attack value of 10
6. **Lives:** The player starts with three lives losing one if out of bounds or killed

The player has the following abilities:

1. **Movement:** The player can use either the left and right or a and d keys to move the player in the corresponding direction with the player sprite switching to that direction as the player moves that direction with a walking animation when currently moving.
2. **Jump:** The player can jump to a little over twice the basic block size by pressing the w or up arrow key once. In addition the player can double jump to reach over four blocks high if timed correctly but expected to make around a three block jump consistently.
3. **Sprint:** The player can hold shift as well as their corresponding movement key to move twice as fast side to side. It should be noted that the shift key by itself does not impact movement but with it held the maximum gap that a player can jump increases from two block gaps to roughly four blocks in length.
4. **Hide:** The player can hide by holding the s or down arrow key to hide behind hideable objects in the world. This prevents the player from being spotted by enemies incorporating the stealth mechanic into our game.
5. **Kill:** The player can kill an enemy by getting close enough to them without being spotted and drawn into combat by pressing space which instantly destroys an enemy.

The player has the following combat options:

1. **Attack:** The player can attack the enemy doing Attack Damage to it with a guaranteed hit and giving the enemy a turn in combat.
2. **Flee:** The player can flee from the enemy with a success rate based upon the players HP with higher HP having a lower chance to flee and a HP below 10% to always succeed at fleeing.

4.2.2 Enemy Characters

The standard enemy has the following abilities:

1. **Detect player:** The enemy can detect the player in a semi-circle based on their view distance and height cap to know if the enemy is in their range.
2. **Start Combat:** Once the player has been detected and was in view for half a second the enemy initiates combat with the player

3. **Turn Around:** Every two seconds the enemy switches what direction it is looking in with the associated visual and detection changes.

The standard enemy has the following combat options:

1. **Attack:** The enemy attacks the player dealing damage in a range from ¾ enemy attack value to 1.5x enemy attack value.
2. **Heal:** The enemy heals themselves recovering between 7 and 20 health points

The basic enemy has a knight sprite with two sprites based upon if it can detect the player or not and a triangular view range to show players where they can be detected. The basic enemy has the stats below:

1. **Health Points:** Starts out with 100 health points but can heal
2. **Attack:** Deals 7 base damage to player based on attack algorithm above
3. **View Distance:** Can spot the player within 400 px hemi-sphere limited by cap below
4. **View Height Cap:** The enemy can only look 3 blocks above itself for the player

The boss enemy has a modified knight sprite with an idle animation that wait until the player is within range to begin a specialized boss combat sequence. The boss's vision is simple distance based with no cone and has the stats below.

1. **Health points:** Has 150 base health points with two health bars, one primary one with 50 points and a secondary one with 100 health points.
2. **Attack:** Deals between 7.5 and 15 damage with each attack
3. **Heal:** Has a limited heal compared to the other enemies of 4-15 health recovery

4.2.3 Other Objects

Hideable objects allow the player to escape enemy detection so long as they are behind the object. Below is a list of hideable object descriptions.

1. **Bush:** Has a moving bush sprite and is the most common form hidedable. Other bush sprites do not move, serving as a distinction for the player.
2. **Barrel:** Is a slightly different color from other barrels being around 30% lighter in color compared to the barrel in the tileset.
3. **Complete_Level:** Invisible check object that sends the player to the next level on contact
4. **Health_Source:** Environmental item the player can consume to replenish their health after combat.
5. **Platform:** The platform is an invisible object that supports the player from falling through the floor as well as providing horizontal blockers that prevent traversal through walls. In addition the platform prevents enemy detections through itself allowing the player to hide behind walls and above on platforms.

4.3 Collectables

As mentioned before, sneaky knight has health potions scattered across the map that will replenish the player's health after they take damage in combat. The player does not have an inventory, so these potions are used immediately after they are acquired.

Section 5 Game Play I/O Controls and GUI

5.1 Game Play I/O Controls

The main gameplay controls are outlined below by key then function.

- **Enter:** Lets the play progress from the title screen
- **Esc:** Pauses the game and brings the player into the settings menu
- **W/↑:** Lets the player jump
- **A/←:** Moves the player to the left
- **D/→:** Moves the player to the right
- **S/↓:** Lets the player couch behind hideable objects; lets the player consume health sources
- **Space:** Lets the player instant kill an enemy
- **Shift:** Increases player movement by a factor of two
- **Mouse Click:** When on a button object allows for combat interactions

5.2 GUI Interfaces

When interacting with the GUI the player has two different options clicking within a predefined box or clicking on an object. For predefined behavior we had the pause menu which would always be static for different settings and as such all interactions are based upon clicking within a pixel area related to the setting. For the object we started with a parent class obj_button. This parent defines clicking behavior with an override function in the create area for children to define what happens upon clicks or hovering.

1. **Attack button:** When clicked turns a global attack variable to true to be processed by the game controller
2. **Flee button:** When clicked, depending on room, do a random check based upon player health to determine if fleeing was successful and pass the results to the game controller to handle. Upon failure show a message on the screen signifying that fleeing failed.

Outside of combat, there is a GUI present for the platformer portion of the game. Although there is no user input from this HUD, it still displays the player score and the number of lives remaining.

Section 6 Visual & Audio Features

6.1 Visual Features

In the main version of the game there are four tilesets used for generic level creation. The tilesets each have a unique theme as outlined below

1. **Outside Foreground:** This tileset is based around an outdoor city or rural town with the main features being a grass or cobblestone floor texture as well as fencing, trees, barrels, and a crate to decorate as needed. Included in the tileset are enough pieces to build a variety of different styles of housing.
2. **Outside Background:** This tileset is based around enhancing the foreground tileset by adding in background characteristics. It allows for the placement of some large mountains in the background as well as clouds that can go into the sky.
3. **Inside Castle:** This tileset is based around the inside of a castle with a thematic door, a variety of similar style platforms, as well as wooden platforms. It can also be used to enhance the Outside Foreground tileset as seen in the later stages of rm_city.
4. **Castle Background:** This tileset is to provide a background for the interior of the castle in conjunction with the Inside Castle tileset. The main features are a pillar, window, and standard brick tile to form a corridor for the player to traverse along as seen in rm_level_1.

Below are some of the object sprites used in the game not related to any specific object.

1. **spr_castle_brick:** This sprite is used to pattern any inside castle backgrounds and was pulled from the Castle Background tileset.
2. **spr_heal:** This sprite is used to denote when an enemy heals in the game.
3. **spr_point:** This sprite is used in addition to the enemy sprite shift to make it clear when the enemy spots the player.
4. **spr_combat_bkgd:** This sprite is used as the background for out combat mini-game made from a combination of other sprites in the game.
5. **spr_splash_screen:** This sprite is what welcomes the player into the game with our splash screen.
6. **spr_square:** Our game took its first breath of life with this humble sprite. The simple 64x64 pixel square defined how large all of our other sprites would be by serving as the foundational piece. It was also used as temporary sprites for every new object in the game. Looking at its children sprites can show where we based much of our more advanced sprites and related code. By having multiple frames or distinct sides we were able to confirm our movement and directional code without the need for the final sprites to be made. In addition it serves as the sprite

for all of our blockers that define the level making it easy to see when the layer is not hidden the limits for player movement.

Section 7 System Parameters & Requirements

7.1 System Requirements

Game Maker Requirements

Gamemaker IDE Version 2023.2.1.75
Gamemaker Runtime Version 2023.2.1.90

Minimum System Requirements

128 MB of Ram
2Ghz+ Processor
64 MB of Graphics
5 MB of Storage

Section 8 Links

Product Backlog (Trello Board):
https://trello.com/w/gamedesign02170129
Git Hub:
https://github.com/ebeckett01/S23GameDesign
Git ssh:
git@github.com:ebeckett01/S23GameDesign.git