Scale the Skyscraper

Game Design Document

Sam Allen

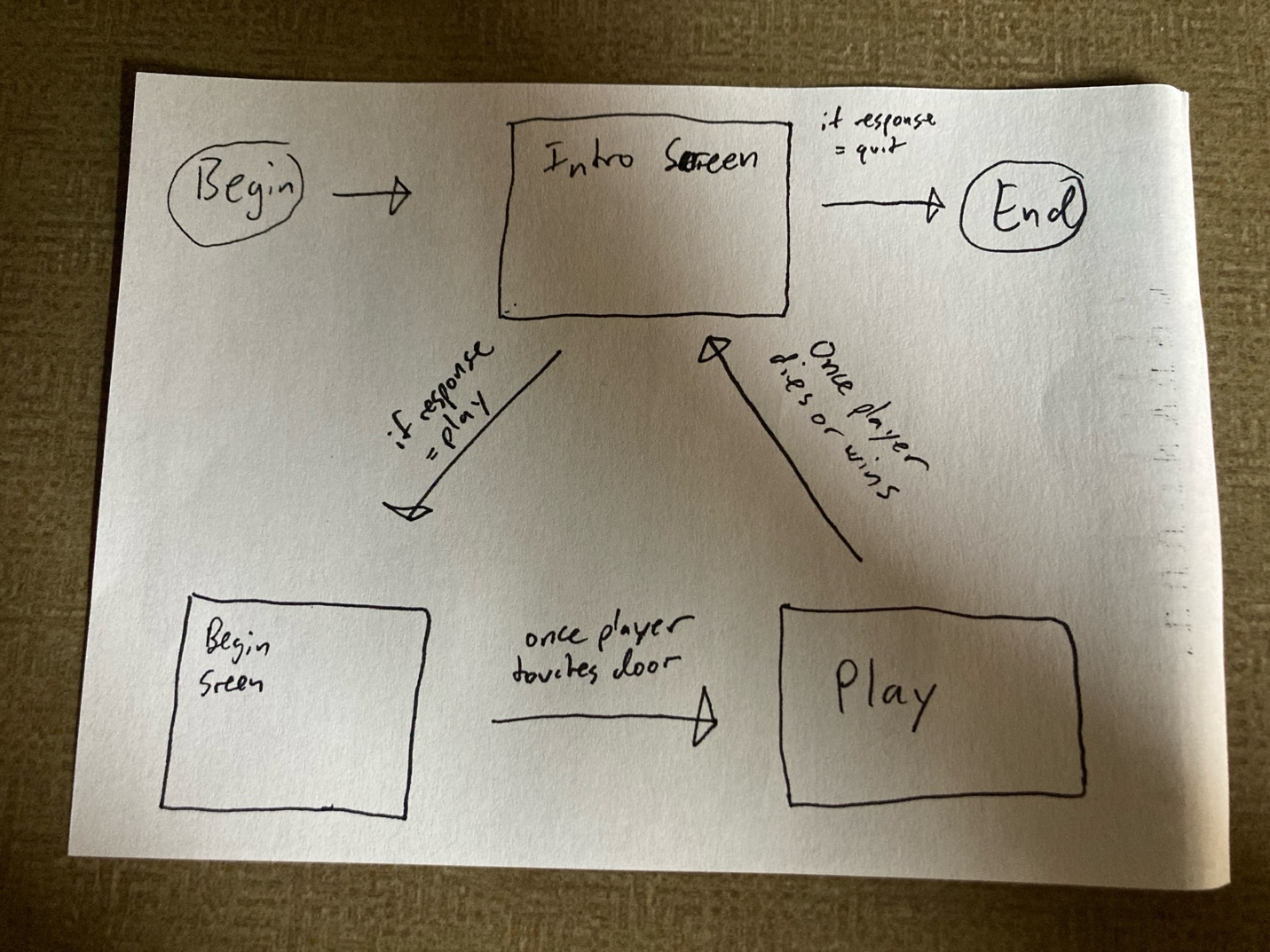
Overview:

Scale the Skyscraper will have a basic 2D game set up, with some adaptations, in order to explore and get a better grasp of simpleGE.

The player will play as Samm, a worker’s rights activist created using photos of myself. It will appear at the bottom of the screen, standing on the ground outside at the bottom of a tower. The player can move left and right with either the arrow keys or the A and D keys and jump using the up arrow or the W key. The player can use the shift or space key to move faster. The player can also use a mouse or trackpad to move the pointer around on screen and interact with objects and enemies. The player will enter the tower, appear in the center of the screen, and then begin to climb, encountering enemies, some of which can be killed by jumping on and all of which can be stopped using the mouse. When the character jumps, the screen will move down to create the illusion of jumping but they’ll stay in the center of the screen the entire time. The game continues until the player makes it to the top and defeats the evil CEO.

When the game begins, it will show an intro screen with instructions and two buttons. The play button will take the player to the “enter” state. Once in the “enter” state the player will walk to the door of the tower, which will take them to the “play” state. If they die or win, they will be taken back to the intro screen, which will show how far they got. If they choose the quite button, it will exit the game.

**State Transition Diagram**



The game will have a 3-state system. Each state will be represented by a subclass of the simpleGE Scene class. The player is first sent to the Intro Screen Scene, which will have instructions, 2 buttons (play/quit), and a slider that shows how far the player got. Both buttons will close the scene and set a variable “response” to whatever the player’s choice was. If they chose to play the game, they will be sent to the Begin Scene. If they chose to quit, the game will end. The player can get from the Begin Scene to the Play Scene by making the character collide with the door on the tower.

The game play scene ends when the player hits and enemy and “dies” or if they beat the final boss. If they died, it will look at the y value of the ground of the tower, subtract half the height of the screen (since the ground starts at the middle of the screen), divide the resulting integer by the height of the tower, multiply that result by the length of the slider, and then add the distance from the side of the screen to the slider in order to get where the sprite of the character should be. If they won, the slider will be replaced with a text box that says “Congratulations, you won!”

**Instructions Scene**

Simple screen that controls access to the game, subclassed from simpleGE.Scene

A paper with writing on it

Description automatically generated

This scene has four main visual elements

* Instructions: a stock simpleGE multilabel containing instructions for game play
* sliderBar: a sprite that resembles the sliding bar showing the previous progress
* btnPlay: a stock button indicating “Play,” leading to the begin state
* btnQuit: a stock button indicating “Quit,” exiting the program
* Samm: sprite that resembles the playable character, used to show how far the player made last time on the slider bar

Other attributes:

* prevProgress- integer variable indicating the y-value of the top of the ground sprite (since it moves downwards as the player climbs the tower), passed into the class initializer and displayed on the sliderBar sprite
* response- a string variable representing the user’s intentions. Set by the two buttons and read in the main function

***Peusocode for instructions scene***

Define a function called determineProgress(prevProgress):

sliderBarProgress gets (prevProgress-half the height of the screen) divided by the total height of the tower multiplied by the length of the slider.

return sliderBarProgress

Create a class called Character:

Samm is a subclass of simpleGE.Sprite

Size should be roughly 60 by 100

Transparent background is preferred

Initial position is slightly above center of the screen

init(samm):

set image to sammFront.png

set size to roughly 60x100

set position to (320, 290)

set moveSpeed to 5

set bounding actions to STOP

process:

If A key or left arrow key is pressed:

Set image to sammLeft.png

Subtract moveSpeed from x

If left shift key or space bar is pressed:

Subtract moveSpeed from x

Else if D key or right arrow key is pressed:

Set image to sammRight.png

Add moveSpeed to x

If left shift key or space bar is pressed:

Add moveSpeed to x

If W key or up arrow key is pressed:

If A key or left arrow key is pressed:

Set image to sammLeftJump.png

Else if D key or right arrow key is pressed:

Set image to sammRightJump.png

Otherwise set the image to sammJump.png

If response is equal to “begin” or response is equal to “bossFight”:

If the variable inAir is False

Set inAir to True (using the game’s process method, this variable will be set back to False when the bottom of the character collides with a platform)

Set dy to -100 (value subject to change)

Use addForce() to add a force of 4 at 270º

\*\*\*The character will be the thing that jumps in these certain rooms, the rest of the room won’t move down. \*\*\*

If response is equal to “begin”

Set bounding action to STOP

Else

If the left side of the character is less than or equal to the left side of the wall

Set dx to 0

If the right side of the character is greater than or equal to the right side of the wall

Set dx to 0

Init(sliderBarProgress):

Set image to outsideBackground.jpg

Set response to “play”

Create instructions multilabel called lblInstructions

Set instructions center to (320, 200)

Set instructions size to (400, 200)

Create a victory multilabel called victoryInstructions

Set victory center to (320, 400)

Set size to (100, 50)

Create btnPlay

Set text to “Play”

Set center to (100, 400)

Create btnQuit

Set text to “Quit”

Set center to (540, 400)

Make a sprite using the sliderBar.png

Set center to (320, 400)

Set size to (160, 30)

Create an instance of the Character class called samm

Set its size to (30, 50)🡨a rough estimate for now

The variable {sliderBarProgress} gets the result of the function determineProgress with the parameter {prevProgress} passed in

Try to Set center to (sliderBarProgress+240, 400)

Try to Add samm, btnPlay, btnQuit, sliderBar, lblInstructions to sprites

If that doesn’t work, check to see if prevProgress is equal to “win”

Add btnPlay, btnQuit, lblInstructions, victoryInstructions

All event-handling will happen in the scene’s process() method

process():

If the quit button is pressed:

Stop the scene

If the play button is pressed:

Set response to “play”

Stop the scene

**The Begin Scene**

Simple screen that allows the player to get used to the controls and get ready to play, subclassed from simpleGE.Scene

A paper with writing on it

Description automatically generated

Begin scene will have visual attributes

* Samm: made from an instance of the Character class (see below)
* door: made from an instance of the Door class (see below)
* cloud: an instance of the Cloud class (see below)

It will also contain some non-sprite assets

* moveSpeed: a variable that determines how fast the player moves
* jumpSpeed: a variable that determines how fast the ground moves away from the player
* roomName: a variable that contains the room name, which is “beginning”
* inAir: a Boolean variable that is False if the character is on the ground and True if the character is in the air
* sndDoor: a stock instance of the simpleGE.Sound class, plays when the character interacts with a door
* sndStop: a stock instance of the simpleGE.Sound class, plays when the character’s mouse hovers over something and stops it

***Pseudocode for begin scene***

Init(prevProgress):

Set image to outsideBackground.jpg

Create an instance of the Character class called samm

Create an instance of the Cloud class called cloud

Create an instance of the Door class called door

Set instructions center to (50, 100)

Set instructions size to (200, 50)

Set instructions text to (Hover the mouse over the cloud!)

Add samm, cloud, door, cloudInstructions to sprites

All event-handling will happen in the scene’s process() method

process():

If samm collides with door:

Stop the scene

Set response to play

Create a class called Cloud

Cloud is a subclass of simpleGE.Sprite

Size should roughly be 214 by 100

Initial position is top left of the screen

Init(cloud):

Set image to cloud.png

Set size to roughly (214, 100)

Set position to (50, 50)

Set dx to -5

Set bounding action to WRAP

Process:

If mouse hovering over cloud is equal to True:

dx = 0

Otherwise dx = -5

Create a class called door

Door is a subclass of simpleGE.Sprite

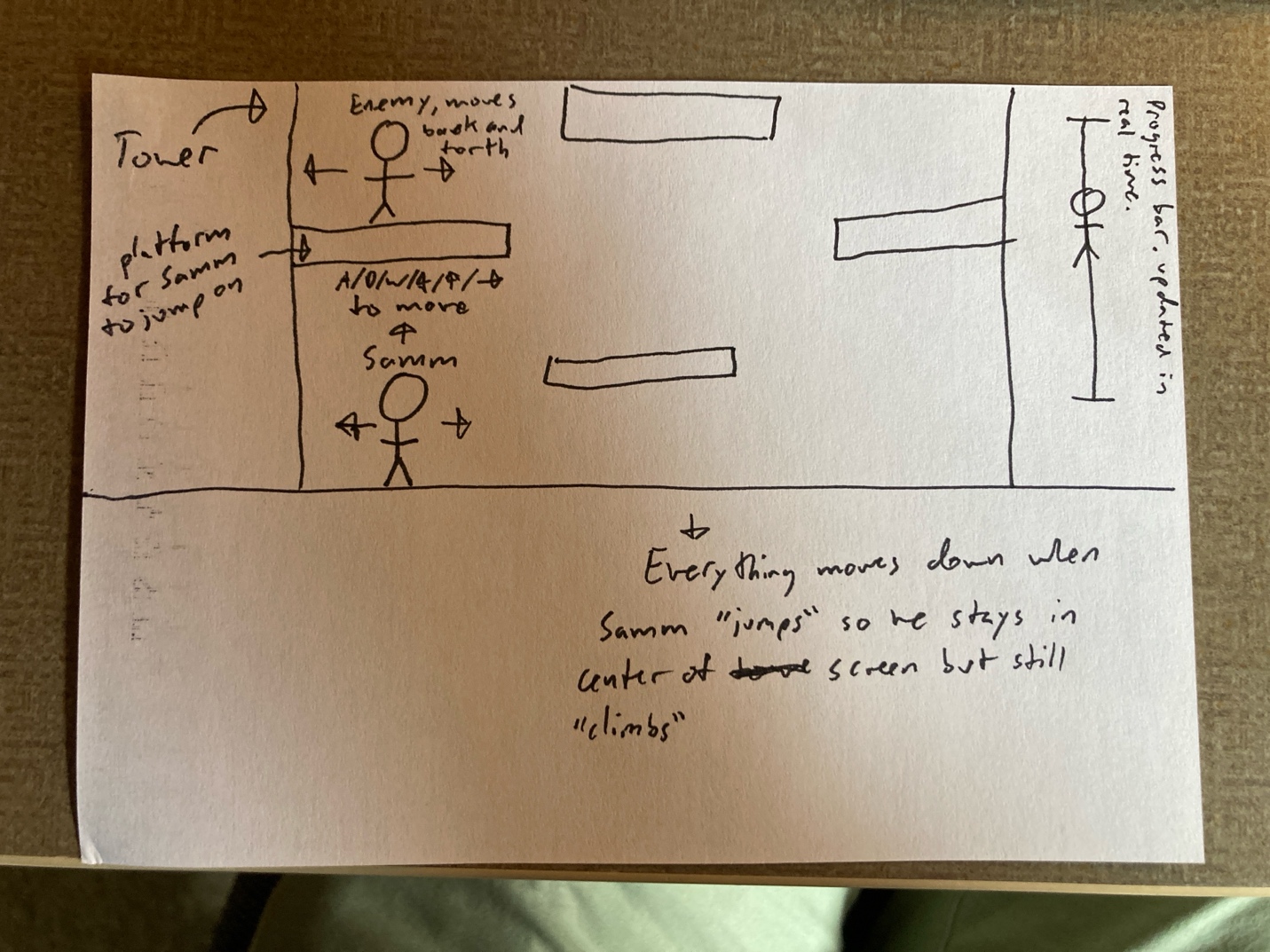
Size should roughly be 60 by 100

Initial position is middle right of the screen, in the building

\*\*\*collision with character will be handled in the scene’s process()\*\*\*

**The Play Scene**

This is the primary scene of the game, subclassed from simpleGE.Scene



Play scene will have more visual attributes

* Samm: made from an instance of the Character class
* Enemies: a list of instances of the Enemy class (see below)
* sliderBarSideways: a sprite that resembles the character showing the current progress, an instance of the Slider class
* ground: a sprite that resembles the ground that the character can stand on, an instance of the Ground class (see below)
* platforms: a list of instances of the Platform class

It will also contain some non-sprite assets

* inAir: a Boolean variable that is False if the character is on the ground and True if the character is in the air
* sndDoor: a stock instance of the simpleGE.Sound class, plays when the character interacts with a door
* sndDies: a stock instance of the simpleGE.Sound class, plays when samm gets killed
* sndStop: a stock instance of the simpleGE.Sound class, plays when the character’s mouse hovers over something and stops it

***Pseudocode for the Play Scene goes here***

Create a class called Enemy:

All of the enemies will be a subclass of simpleGE.Sprite

Size should be roughly 40 by 80

Transparent background is preferred

Initial position will be the position of a platform plus 40 (40% chance one will spawn on each platform)

A variable called originalX gets each enemy’s starting x value

init(enemy):

set image to enemyRight1.png

set size to roughly 60x100

set position to (320, 290)

set moveSpeed to 5

process:

if mouse hovering over the enemy is true:

set enemy’s dx to 0

else if enemy’s x position is greater than or equal to originalX plus 64

set enemy’s dx to negative moveSpeed

else if enemy’s x position is less than or equal to originalX minus 64

set enemy’s dx to moveSpeed

\*\*\*the enemy colliding with the main character and killing them will happen in the scene’s process()\*\*\*

Create a class called Platform:

All of the platforms will be a subclass of simpleGE.Sprite

Size should roughly be 128 by 10

Initial position will be determined in the scene’s process(),

Process()

If the player clicks the up arrow or W

If inAir is equal to false

Set inAir equal to true

Set dy to 100

If inAir is equal to True

Use addForce() to add a force 4 at angle 90º

Init(prevProgress):

Set image to insideBackground.jpg

Make a sprite using the sliderBarSideways.png

Set center to (590, 240)

Set size to (30, 160)

Create an instance of the platform class called ground, set size to (640, 240), set center to (320, 360), set image to ground.png

Create an instance of the Character class called samm1

Set its size to (50, 30)🡨a rough estimate for now

The variable {sliderBarProgress} gets the result of the function determineProgress with the parameter {prevProgress} passed in

Set center to (590, sliderBarProgress+40)

Add samm, samm1, door to the sprite list

For i in \*\*\*y-value of highest part of the tower below the boss room\*\*\* to 240

If i/50% is equal to 0

Make an instance of the enemy class with y value equal to i +40

Add that instance to the list of sprites

For i in \*\*\*y-value of highest part of the tower below the boss room\*\*\* to 240

If i/50% is equal to 0

Make an instance of the platform class

Generate a random number between 1-10

If that number is less than or equal to 4 (this is intentionally slightly less likely)

Set the platform’s center to (184, i-10)

If that number is greater than or equal to 5 (this is intentionally slightly more likely)

Set the platform’s center to (420, i-10)

Add that instance to the list of sprites

For i in \*\*\*y-value of highest part of the tower below the boss room\*\*\* to 240

If (i+25)/50% is equal to 0

Make an instance of the platform class

Set the platform’s center to (320, i-10)

Add that instance to the list of sprites

All event-handling will happen in the scene’s process() method

process():

Set center of samm1 on the slider bar to (590, sliderBarProgress+40)

If samm collides with ground

Set inAir equal to false

If samm collides with the top of a platform

Set inAir equal to false

If samm collides with door

Stop the scene

Set response to “bossFight”

If samm collides with an enemy

Play dies.wav

Stop the scene

Set response to “instructions”

**Boss Fight Scene**

A paper with a drawing of people and text

Description automatically generated

Button

Moves toward the player using force() so it has to build up speed and slow down. Jumps randomly. Each platform has a button, if they all get pushed within a time limit, the boss dies

Button

Button

Button

If all buttons get pressed, boss dies

Boss Fight Scene will have other visual attributes

* samm: made from an instance of the Character class
* boss: made from an instance of the Boss class
* platforms: a list of instances of the Platform class
* buttons: a list of instances of the button class

It will also contain some non-sprite assets

* sndBonk: a stock instance of the simpleGE.Sound class, plays when samm jumps on the boss’s head and damages it
* sndDies: a stock instance of the simpleGE.Sound class, plays when samm gets killed
* sndSlows: a stock instance of the simpleGE.Sound class, plays when the character’s mouse hovers over the boss and slows it down
* variable buttonsPressed is equal to zero, if all four are pressed, the character wins

***Psuedocode for Boss Fight Scene will go here***

Create a class called Boss:

Boss is a subclass of simpleGE.Sprite

Size should be roughly 100 by 150

Transparent background is preferred

Initial position is slightly center right of the screen

init(boss):

set image to boss.png

set size to roughly 100x150

set position to (480, 240)

set bounding actions to STOP

process:

if the x coordinate of the character is less than the boss

if mouse hovering over the boss is False

addForce 3 at 180º

if mouse hovering over the boss is True

add force 2 at 0º

if the x coordinate of the character is greater than the boss

if mouse hovering over the boss is False

add force 3 at 0º

if mouse hovering over the boss is True

add force 2 at 0º

if the left side of the boss is less than or equal to the left side of the building wall

set dx to 0

if the right side of the boss is greater than or equal to the right side of building wall

set dx to 0

If mouse hovering over the boss is false

if the boss’s x coordinate is within 310-330

if bossInAir is equal to False

set bossInAir equal to True

set dy to -15

if y coordinate of the bottom of the boss is greater than the y coordinate of the top of the floor/platform

addForce 3 at 270º

if the y coordinate of the bottom of the boss sprite is less than or equal to the y coordinate of the top of the floor/platform

set bossInAir equal to false

\*\*\*collision between character and boss will happen in the scene’s process()\*\*\*

Create a class called Button:

Button should be a subclass of simpleGE.Sprite

Size should be roughly 30 by 20

Transparent background is preferred

Initial position is determined by the position of each platform

init(button):

set image to button.png

set size to roughly 30x20

\*\*\*handle collision between character and button in scene’s process()\*\*\*

Create the room

Init():

Set image to bossRoom.jpg

Create an instance of the Character class called samm

Create four instances of the platform class, manually set centers to (320, 100), (320, 340), (540, 240), (540, 240)

Create four instances of the button class called button, manually set centers to above each platform

Create an instance of the Boss class called boss

Add samm, platform x4, button x4, boss to sprites

All event-handling will happen in the scene’s process() method

process():

If samm collides with boss:

Stop the scene

Set response to instructions

If samm collides with button:

Change photo to buttonPressed.png

Add 1 to variable buttonsPressed

if buttonsPressed is equal to 4

play victory.wav

stop the scene

set response to instructions

set prevProgress to “win”

***Psudocode for main() function will go here***

Will handle transition between states

Uses a main loop, with \_\_\_\_ variables

* instructions: an instance of the Instructions class
* begin: an instance of the begin class
* play: an instance of the Play class
* bossFight: an instance of the BossFight class

main():

set keepGoing to True

prevProgress = 0

while keepGoing is True:

Create an instance of instructions

Pass the previous progress in as a parameter

Start instructions

When instructions ends

If instructions.response is “begin”

Create an instance of begin

Start begin

If instructions.response is “quit”

Set keepGoing to False

If begin.response is “play”

Create an instance of play

Start play

If play.response is “bossFight”

Create an instance of bossFight

Start bossFight

If play.response is “instructions”

Create an instance of instructions

Start instructions

If bossFight.response is “instructions”

Create an instance of instructions

Start instructions

***Milestone plan***

1. ~~Get begin scene working~~
2. ~~Get samm, cloud mechanics working~~
3. ~~Get play scene working~~
4. ~~Get enemy, ground, and platform mechanics working~~
5. ~~Get prevProgress working correctly~~
6. ~~Get bossFight scene working~~
7. ~~Get boss sprite mechanics working~~
8. ~~Get button mechanics working~~
9. ~~Get instructions class and state transition working~~
10. ~~Get sounds fully working~~

**Asset plan**

insideBackground.jpg

A close up of a white surface

Description automatically generated

outsideBackground.jpg

A drawing of a city with trees

Description automatically generated

bossRoom.png

A red rectangular object with black lines

Description automatically generated

characterWalk.png

A collage of a person standing in different poses

Description automatically generated

sliderBar.png

A green arrow with a star in the middle

Description automatically generated

slideBarSideways.png

A green arrow with a white background

Description automatically generated

cloud.png

A grey cloud drawing on a white surface

Description automatically generated

door.png

A drawing of a door

Description automatically generated

enemyWalk.png

A screenshot of a video game

Description automatically generated

ground.png

A drawing of a skeleton on a piece of wood

Description automatically generated

platform.png

A green rectangular object with a black border

Description automatically generated

button.png

A red and white rectangle

Description automatically generated

buttonPressed.png

A red and black rectangle

Description automatically generated

sndDoor.wav

slows.wav

sndStop.wav

dies.wav

button.wav