Michael Zhang
Intro to Game Engineering: Final Project

Useful Links:

- **README**: https://github.com/mwz210/CS3113-HW-Project/blob/master/Final%20Project%20(2D%20Platformer%20Contra)/README.MD
- **Basic controls**: https://github.com/mwz210/CS3113-HW-Project/blob/master/Final%20Project%20(2D%20Platformer%20Contra)/Basic-Controls.txt

---

Shitty Contra: the most average and unremarkable platformer -- is what anyone looking at the game thinks on first glance.

With level one being a grassy level, level two being a cave level, and level three being a boss fight, Shitty Contra does, indeed, seem to be the generic and formulaic platformer.

But is it?

Enter a platformer that's ridiculously hard for all the wrong reasons. Get stuck in the wrong place? Random solid-looking blocks are slippery? Bouncy springs to certain ledges seem more like springs to death? Screen shakes like you're flying up Kingda Ka? If these features sound like your cup of tea, *Shitty Contra* might be the game for you.

Shitty Contra was originally based on the platformer *Contra*, but as I kept creating the game, the more and more it diverged. I wanted it to be ridiculous and keep the audience on their toes, and as I playtested it on friends, their reactions pretty much summed up how I wanted the user experience to be -- "What the heck?" "WHY CAN'T I GET OUT?" "Oh look! It's so cute!--" (referring to the boss) -- "WAIT WHAT IT KILLED ME!"

Here's how I designed this game:

Game Flow and Navigation: I used the finite state machine design to create the design flow for the game, running the title screen first, then Game Levels 1, 2, and 3 sequentially. If the player dies, they go to the "Game Over" screen and have to start all the way from Level 1. In practice, something I still need to work on in the game is the lag. It doesn't really lag when running individual levels, but with the FSM design, Level 2 is very laggy. Thus, please read the README on Github for instructions on running the individual levels.

Level 1: Grassland

The features I implemented in Level 1 were:

- Bouncy spring objects that launch the player high into the air. These are necessary to get through the level, and are pretty difficult to spring off correctly, making for some funny moments. I used point-box collision between the springs and the player, and the box-bounce mechanism to make the springs.

- A simple enemy entity, reminiscent of the gumbas from Mario, that moves left and right and doesn't walk over edges. I used point-box collision for the enemies with solid blocks, and checked for edges with point collision.

- The player can shoot bullets to kill the enemies. I did this by initializing all the bullets far away from the screen. The player can shoot both left and right. When the bullet gets far enough, it will be set to somewhere far away from the screen so the player may use it to shoot again (at one time the player can have 5 bullets on them).

- The player and the enemies both die in an explosion if either die. I did this by creating particle and animation effects for both player and enemy; the player death sets off a particle emitter that's attached to it, while the enemy death sets off a different animation that resembles an explosion.

Level 2: Cave

The features I implemented in Level 2 were:

- Instead of having enemies in this level, the player navigates through the level by jumping on a sequence of small blocks, making this level more technically challenging than the last. If the player falls into the spikes on the bottom, they die -- I implemented this by having the player die below a certain y-coordinate. Also, since there are no enemies, the player doesn't have the ability to shoot in this level.

- In this level, the player has to retrieve a key near the bottom of the map to unlock the door to the next level. To do this, I created objects for both the door and the key. When the key collides with the player, I changed a boolean variable inside the player to acknowledge that the player has the key. Then, if the player has the key and they collide with the door, I move the object for the door to somewhere far beyond the screen, so it effectively disappears. If the player doesn't have the key, then the door behaves like a solid object.

- Whenever the player jumps, the screen shakes, effectively making it very disorienting to jump. I did this to make the level unexpectedly hard, but people with motion sickness may not enjoy playing it. In the future, I will work to tone down the screen shaking. Right now, I implemented it using the sine function on the view matrix at the x position.

Level 3: Boss Battle

Those who made it to the boss battle are probably wondering what the cute and friendly boss can do in comparison to the previous levels. When you first enter the level, the stationary boss, like a pink Diglett, is sleeping inside the ground.

However, I implemented the boss to have three modes using animations: the sleeping mode, the normal mode, and the rage mode. When it is sleeping, the player can't do any damage to the boss. When the boss is in normal mode, they don't do anything to the player even if the player shoots it, and the boss is in this mode when it's above a certain amount of health. However, if the player gets too close, or if the player reduces its health enough, the boss enters the "rage" animation and starts firing a stream of bullets.

I implemented all this by attaching bools to the boss and updating them according to the player's distance from the boss and the boss's health level.

Finally, if you manage to beat the boss, a particle emitter is set off, and the screen will shake. Well done, and congratulations! You made it through *Shitty Contra!*