End of Term Status Presentation

Group 2 - 2D Metroidvania

Cristino Lamadrid, Karen Salinas,
Keith Harryman, Jose Reyes

Overview

1. Doorways and Saving System
2. Enemy AI and Crawler Enemy
3. Camera Movement Scripting and Weapon System
4. Texture Tiling and Minimap

Goals

- Player movement controller script
- Enemies and scripted AI
- Camera manipulation
- Level design (background tiling and parallax)
- Level saving and loading
- Event Manager

Doorways

Transports the player to either a new scene or a different part of the current level.

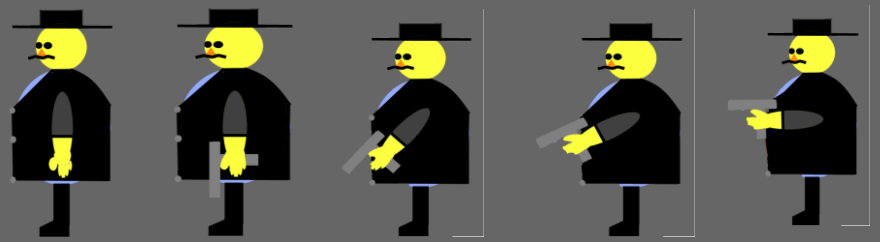
Save/Load Game

- Saving the game
  - Move all of the game state that we care about into a C# object.
  - Serialize into a binary file in the filesystem.
- Loading the game
  - Read the saved file and deserialize into a C# object.
  - Move that state back into our game objects.
  - Start the game.


Restart

Sprites Used

Enemy CEO



Oil

Enemy Scripted AI

Enemy shooting animation :

- Consists of four sprites of the enemy grabbing a gun from his pocket, then raising it up and firing during the last clip.
- It has a “firingTime” and “notFiringTime” to delay bullets spawning, and allow enemy walking movement. Another time, fireTime, is time which prompts when to spawn the bullet after the firing animation begins:

```
animator = GetComponent<Animator> ();
   RuntimeAnimatorController ac = animator.runtimeAnimatorController;
   for(int i = 0; i<ac.animationClips.Length; i++) {
      if(ac.animationClips[i].name == "ceo_shooting"){
             fireTime = ac.animationClips[i].length;
      }
   }
```

Enemy Scripted AI

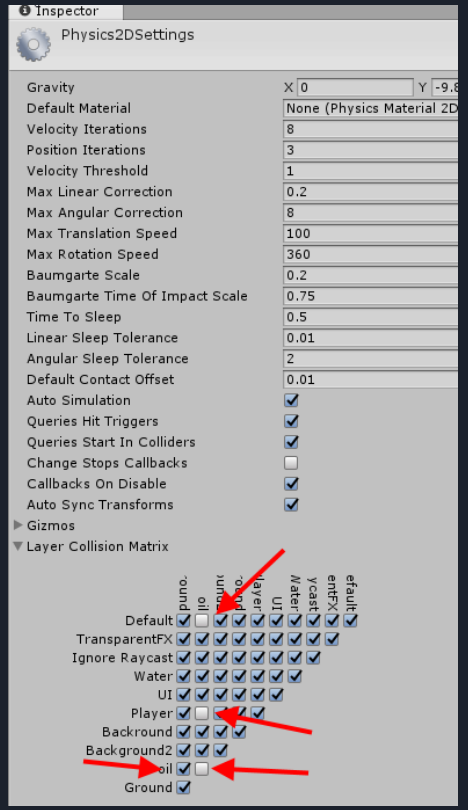
Oil movement :

Consists of two conditional blocks in collision event and two conditional block in update event with two conditional variables, “direction” and “is_right”. The “direction” can be assigned as “horizontal” or “vertical”, and “is_right” is boolean. The “MiddleGround” game tag is used between the two walls, and the regular “Ground” tag is used on the outsides to differentiate between moving back up or moving left/right when the oil moves down to collide with the ground.

Oil Collision Problems & Using Collision Layers

Because of problems with the oil colliding with the player and other enemies, an option provided by Unity was used which can make objects transparent to other layers. By deselecting the Player and Oil layers in the Physics2D options panel, collision will not happen between the oil and other oil objects and between objects in the Player layer which includes the Enemy(CEO) objects.

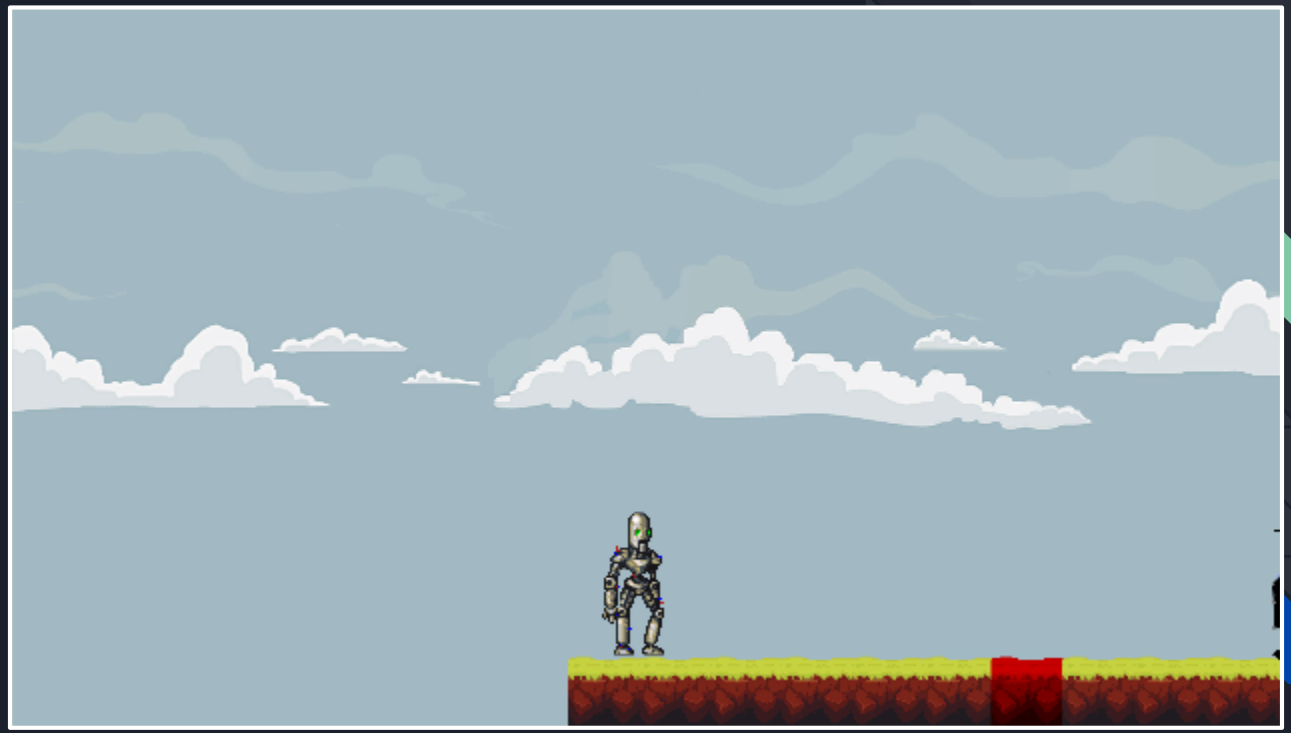
Edit->Project Settings->Physics2D



Camera Follow

- Simple camera follow script can work but is too exact.
- Use a “bounding box”
  - Camera only moves when hitting the edges of this box.
  - Using a hidden Rect object, margins, and max/min values for this box.
  - Linear Interpolation (player.x, camera.x, Time.deltaTime * smoothX)
  - Moves camera to the player position over time with an element of smoothing to keep the camera from moving too fast.

Camera Follow



Player Weapon System

- Requires a GameObject to be instantiated from a prefab.
- Must initially apply the player's velocity to the bullet.
  - If player is moving, they shouldn't be able to catch up to the bullet.
- Apply the expected force to the bullet object.
- Aiming requires taking the mouse location and applying the correct rotation.

Things to consider

Camera Follow

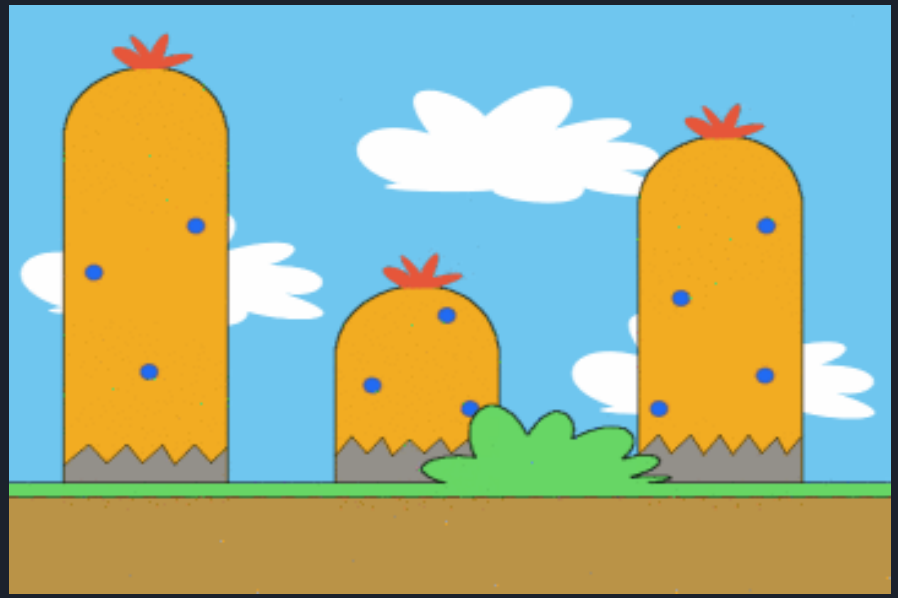
- Implement a method of allowing an external function to hijack the target of the camera follow script. This allows us to show the player a special event such as a new path or door opening.

Weapon System

- Use an event manager to allow the player to easily toggle between weapons or weapon states. This can be passed on to other similar functions such as equipped items.
- Bullet state when the player is moving and aiming at angles or vertically.
- Player animations and sounds.

Tiling

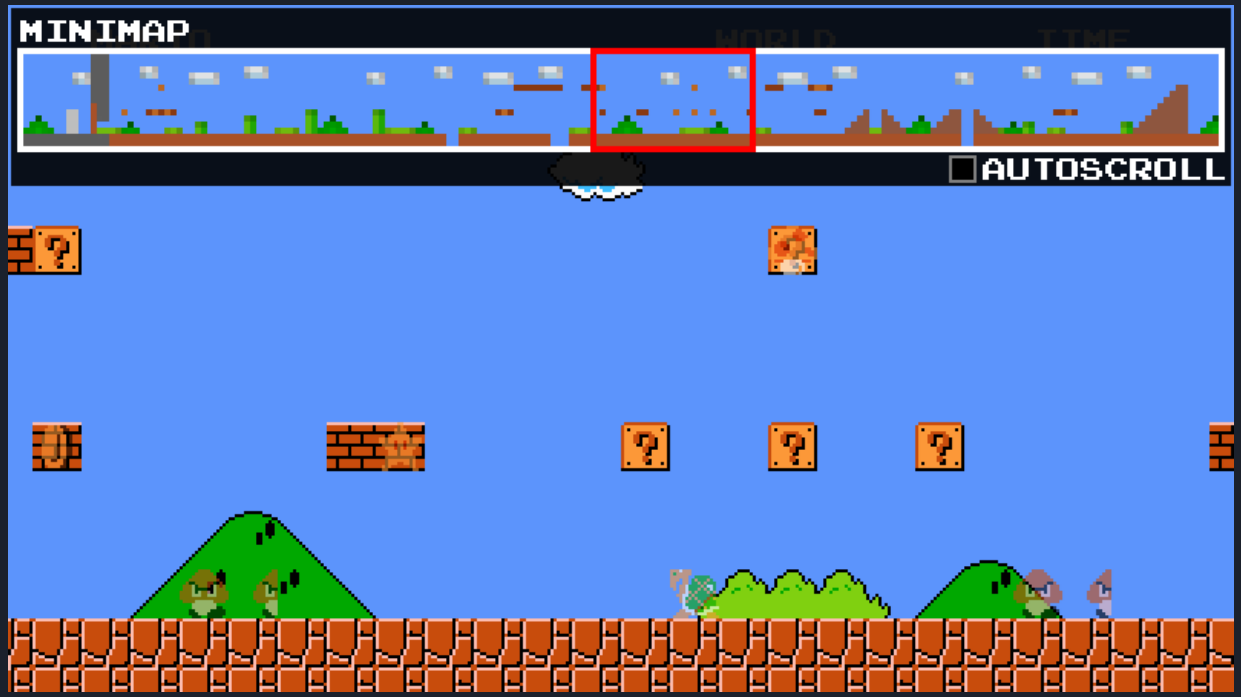
- Makes the foreground and background elements procedurally tile giving the player the ability to infinitely explore the level.
- Helps prevent too much obvious repetition as the same asset scrolls indefinitely.



Objective: Create a Minimap

- Minimaps (or radars) are known for displaying information the player's surroundings.
- Mini-maps should be centered on main character. Then, it should use readable symbols instead of real models because minimaps are often quite small and the player wouldn't recognize the information that the minimap is trying to present.
- Oftentimes, minimaps are circular, but since this is a 2D platformer, we will make it a long strip sitting on top.

Objective: Create a Minimap



Example of a 2D minimap

Goals for next Semester

1. **Animations**
2. **GUI/HUD**
    a. Player Health / Armor
    b. Ammo
    c. Etc
3. **Dialogue System**
    a. Item Pickup
    b. Ability Unlocked
    c. Etc