Epicalyx Studios Final Report for *The Last Cake on Earth*
5/3/2022

Team Member Contributions

Lucas Colegrove:

Lucas worked on creating several UI elements for the game. These elements include the main menu, the pause menu, the death menu, and the inventory system UI. All the game logic for transitioning between the menus and accessing the inventory was handled by Lucas as well. Lucas also assisted in the creation of items and item pickup for the project. Lastly, Lucas created a system which allows the player to obtain items from chests in game.

Jonathan Dolbee:

Jonathan worked on the enemy state machine, enemy animations, and attacks. He also worked on the inventory system and made it dynamic so any item could be placed in any slot. Along with this, Jonathan helped fix bugs related to animation and attack systems. Jonathan also modified the scene transition system to transfer all objects attached to the player object and prevent duplicate players in a scene. Jonathan also helped debug and fix many other systems around the game.

Tia Edora:

Tia made several art assets including the walking and attacking sprite animations for minor enemies, boss enemies, and the main character. Several effect items in the game and the end card were also made by Tia.

Robert Fahey:

Robert designed all of the game environments (three open game areas, the bakery, and two dungeon interiors. Robert designed logic for doors that move the player and menu objects attached to the player between scenes. The base heads-up display implementation, the oven menu, the pause menu, and the endgame screen were all designed by Robert. Robert worked with Lukas to wire the teleporter system in the dungeons. Two additional tile sets and all of the game music were chosen and imported by Robert. Robert conducted the final playesting session prior to the final demo to ensure the game could be properly beaten and the demo fits within the allotted time limit.

Lukas Frick:

Lukas created the logic for random dungeon traversal. He also added most of the sound effects, and helped with setting up the background music to work with the rest of the game's sounds. Lukas also assisted in the implementation of the player animations and the state machine that governs said animations. Lukas worked on the final adjustments and debugging prior to

submission. These adjustments include adjusting enemy/loot placement, small visual tweaks to tilemaps, adjusting player attack values, and boss hitboxes.

Sam Jones:

Sam designed the state machines for player actions and worked on the underlying implementation of player movement and attacks, as well as implementing the base classes Entity and Enemy and corresponding basic functionality. Sam was also responsible for implementing cooperative multiplayer, but was not able to finish by the May 3rd due date. Sam also created and maintained the team website.

How to Play

Game controls:

- WASD: move the player up/down/left/right
- Left mouse button: attack
- F: use parts to repair the oven
- Q: interact with chests, stairs, and the oven
- E: show/hide the inventory

The player can pause and unpause the game using the buttons at the top left of the screen. These controls are displayed on the pause menu along with an overview of the player's objective. The player can also exit to the home menu and close the game from the pause menu.

The player's inventory has nine slots that contain stacks of items. The player can click on status items in those slots to use them. Parts, while they occupy inventory slots, cannot be used by clicking on them.

Rules and Mechanics

The player's goal is to collect two ingredients (milk and flour) to bake the last cake on Earth. The player collects the ingredients by defeating the bosses in the two factories placed around the city.

Combat:

The player has 10 units of health. The player, enemies, and bosses all deal damage by melee attack. Enemies and bosses will chase the player and attempt to deal damage if the player walks within the enemy's attack range. Slimes have three units of health and are the slowest enemies. Fries have ten units of health and move faster. The two bosses (the mutant slime and mutant cow) have larger attack ranges and deal more damage. Defeating the mutant slime gives the player flour and the mutant cow gives the player milk.

Items:

Items appear in chests placed around the game environment. Two items are implemented: coffee and chocolate. Chocolate heals the player. Coffee heals the player and temporarily increases the player's movement speed. Also placed around the city are parts, items used to fix the oven used to bake the final cake. The player has nine inventory slots. Each inventory slot can hold a stack of up to twenty items of a single type.

Baking the Cake:

Inside the bakery (the open building at the center of the central game area) is the oven. In order to unlock the oven, the player must repair it using 20 parts. Repairing the oven will allow the player to access the oven menu. From that menu, the player can bake the cake if the player has all of the ingredients. Baking the cake shows the player a unique end card and gives the player the option to return to the main menu.

Inventory System:

The inventory system was coded to allow various amounts of items to be placed in it. The system allows you to stack items in each slot with a defined maximum. The item types must match for stacking to work. Once the player has filled an item slot, then any new item picked up will proceed to take an additional slot, until it is also filled up, and so on and so forth. If the player picks up a different kind of item, that item will also take up an additional item slot. For example, if the player picks up two chocolate bar items, their inventory would consist of one slot taken up by two chocolate bar items. If they pick up a coffee item, the inventory would consist of two slots taken, one by chocolate, and the other by coffee.

Design Trade-offs

The overall scope of the game was reduced for the sake of time. The number of dungeons, bosses, and ingredients was reduced to two from the originally planned four. The number of unique enemy types and status items was reduced. Implementation for ranged combat and weapon switching was started, but removed for ease of testing. The original plans to have enemies drop items and to have chests containing multiple items was modified for balance purposes. Instead, items can only be found in chests, and chests drop a single item. More limited item spawns prevents the game from being trivially easy. The hidden cafes originally planned to hide loot chests were not implemented. Instead, chests appear in alleys in the game world. This encourages the player to take different paths through alleys and explore more, instead of sticking to main roads looking for entrances to buildings. Extra bakery features like weapon and storage upgrades were scaled back for the sake of time to allow development to focus on the ingredient/oven implementation.

Playtesting

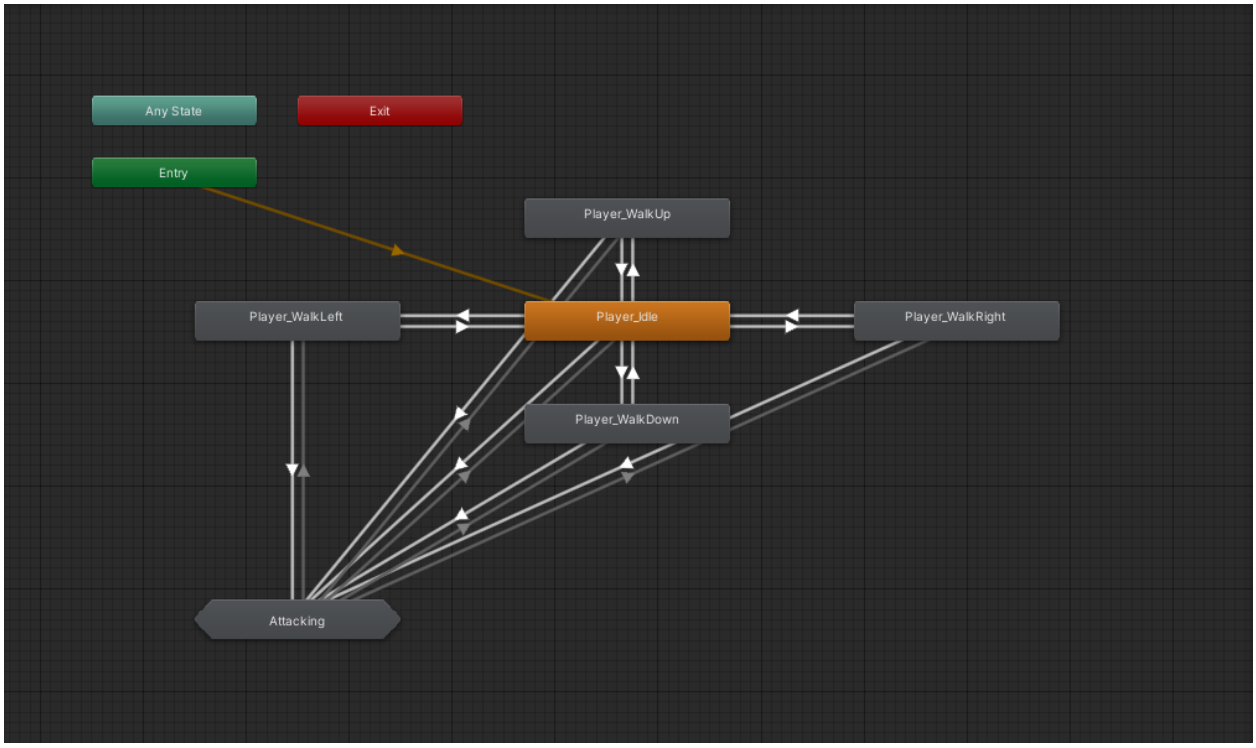
Playtesting revealed that the original balance of the game's combat was too difficult. A combination of the player's small attack range and enemies' movement speed made landing attacks without taking damage too difficult. In response, enemy movement was slowed such that the player could successfully outrun them. The player's attack radius was also increased. Placement of chests was altered to supply the player with enough healing items to have a better chance at completing the dungeons. The number of enemies placed in dungeons was also increased, as the original layouts felt too "empty".

Software Design and Implementation

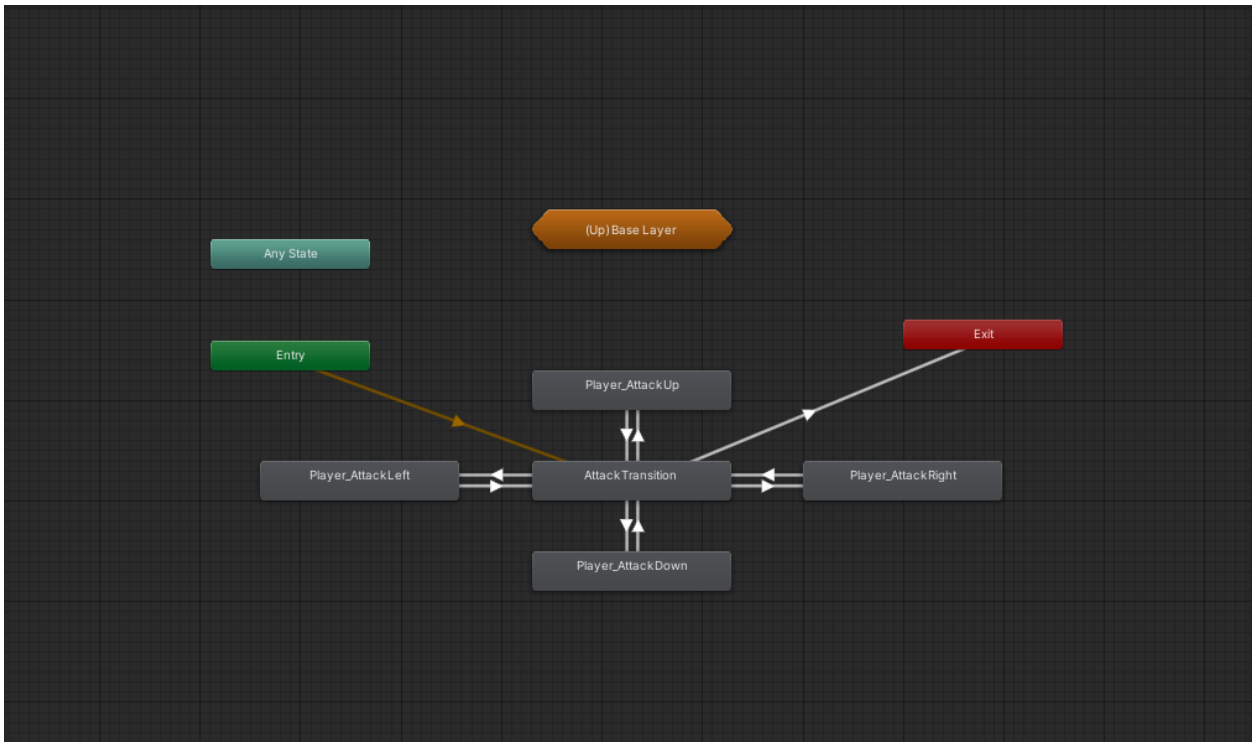
Player Movement and Animations:

The player has an overarching idle state that plays the idle animation. Should the player begin moving in any direction, the current state changes to the matching movement state, which in turn plays the matching walking animation. From any of these movement states or the idle state, if the player attacks, it will transition into an attacking substate machine that will detect the direction the player's mouse was pointing when the attack was triggered. Using this direction, the state machine will transition into the corresponding attacking state, and play the correct attack animation. This attack animation will play over any of the other animations. Once the attack animation is finished, the state machine will go back to the idle state, or transition into a movement state should the player still be moving.

Main player animation state machine:



Player attacking sub-state machine:



Enemy Behavior:

Enemy ai was coded using a state machine. The enemy is constantly in an idle state where animations are played that show it is in an idle state. We then decided on a few more states for the enemy: alert, chasing, and attacking. The alert state is activated any time a Player game object comes into the enemies collider box. This simply lets us code anything we'd want for the enemy to interact with the player while not actively chasing. Then we set a chase range that the enemy state is constantly observing once you are in its alert range. When you cross the chase range boundary, the enemy will begin chasing you. Upon coming into a predefined closer range the enemy will start attacking. When you leave an enemy's chase range, the enemy state machine will return to idle, which then tells the enemy to return to its starting position.

Animation System:

The animation system was designed with the difficulty of adding additional animations for attacks in mind. This was one of the reasons the scope of our project had to be reduced in terms of animation. We settled on two variables to decide animation states. Direction and isAttacking. The direction variable was a number between -2 and 2. -2 being left, -1 being down, 0 being idle, 1 being up, and 2 being right. These variables helped us to build our animation state machine efficiently and quickly.

Environment Design and Navigation:

All environments were implemented as prefabs consisting of layered tilemaps. Solid boundaries are tilemaps with 2D tilemap colliders attached. Each game area is implemented in its own scene. Doors between areas are implemented similarly to the solid boundaries. On collision, the doors load the destination scene, transfer the player and the menu objects attached to the player from the source scene to the destination scene. The destination scene is then set as the active scene.

The two dungeons (the milk factory and the flour factory) consist of a starting room, a boos arena, and a 3x3 grid of rooms (three floors with three rooms each). Rooms are connected by teleporters styled as staircases. The rooms in the grid have both an entrance teleporter and an exit teleporter. Attached to the player is a script that randomly generates a seed for a path through the dungeon. The teleporters read the seed for a room number and send the player to the corresponding room of the next floor.

UI Design:

UI was designed with simplicity and ease of use in mind. All the menus in the game have buttons with clear labeling to help the player navigate the game. The only exception to this rule is the inventory where the player clicks on an item image rather than a labeled button. However, this system was deemed player friendly by people who tested the game. In terms of aesthetic choices, most of the UI was created using pastel colorings of purple, pink, and white. The idea was to give the user an experience that would remind them of a setting found in a typical bakery or cake shop. Button mappings for interacting with in-game objects are similar to what you would find in many games of current day.

Sound Design:

Game music consists of individual tracks attached to the root environment/menu prefabs in each scene, set to loop continuously. Sound effects are set to trigger when their corresponding events occur. Sound effects are set to be able to overlap with the game music. Sound effects exist for footsteps, player attacks, the player taking damage, enemy attacks, enemies taking damage, unique idle noises for each enemy type, and for opening chests.

Sources

All three tile sets used to construct the environment were obtained from Kenney Game Assets at the links below, under the Creative Commons 0 license. Links are as follows:
https://www.kenney.nl/assets/roguelike-modern-city ,
https://www.kenney.nl/assets/roguelike-caves-dungeons ,
https://www.kenney.nl/assets/roguelike-indoors

All music was obtained from the Unity Asset Store at the following link, under the Standard Unity Asset Store EULA.
https://assetstore.unity.com/packages/audio/music/8bit-music-album-051321-196147

All sound effect assets were obtained from Freesound.org under the Creative Commons 0 license.

Team Website Link
https://bob737373.github.io/Epicalyx-Studios-Website/