Phase 4 Report

At the beginning of the project, we had a collection of ideas of how our game would play, but as we worked through phase 2 we made a group decision to implement the game exactly as described in the phase 1 requirements. In addition to the game mechanics, we added a plan for navigable menus that we fully realized once we learned about the Java Swing and AWT libraries.

In our game, the player controls a raccoon gathering apples around SFU. Each level has a set amount of apples that the player must collect before moving to the finish position in order to win. The apples award 10 points each, and there is an optional collectable student ID that awards 30 points that appears in a random location in the level and disappears after 40 ticks. There are snakes that deduct 10 points and disappear on contact, and a moving enemy that causes the player to lose the game on contact. The player will also lose the game if their score becomes negative or when they are out of time. Most of our game arts are custom-drawn (eg: the Academic Quadrangle Courtyard as the background of level 2, the student ID as the bonus, etc.) to capture the spirit of SFU.

One of our original designs was to make the raccoon move continuously by using a fast tick rate and small movement distance, but we decided to follow the grid coordinate system for movement positions outlined in the requirements. The tick rate we settled on was 0.75 seconds. There was some ambiguity in how to design the player movement around the tick rate. We debated between allowing the player to move immediately on a key press once every tick, or to track the last key pressed and perform the movement once the tick occured. We settled on the latter.

A change we made to the requirements was to put in only one moving enemy. Due to the movement mechanics, the player could never gain distance on a moving enemy in an open space, so putting in more than one would make a level nearly impossible to complete. In addition to that we also decided to create an enclosure in each level where the player would be rewarded by strategically trapping the enemy inside. If the direction the enemy wanted to move in is blocked by a wall it would instead stay put, so there was no extra code needed to implement this.

There were optional features and game mechanics we planned that we decided not to implement due to time constraints. This included an item to deter the moving enemy and a key rebinding option.

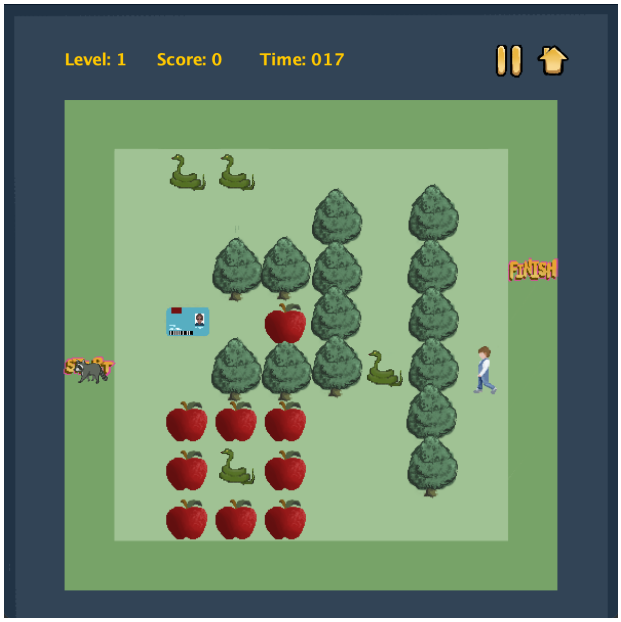
We also performed various code refactors that polished our system design from the original. The entity hierarchy was reduced by removing abstract classes for enemies and rewards and using the stationary enemy and basic objectives directly as super classes for the moving enemy and bonus objectives respectively. We also thoroughly encapsulated all primitive data types in object classes, and the game loop in its own class as well.

As we were all new to the software development process, we learned a lot from implementing this game. We did not start with a thorough planning phase and collaborated loosely to develop the game. We only started adhering to an explicit design (the requirements) part-way into the process with the consequence of much wasted work, and met with quite a few merge conflicts as well. Having some dedicated meetings for planning would have been beneficial, but practical constraints such as differing schedules and course work loads, and no

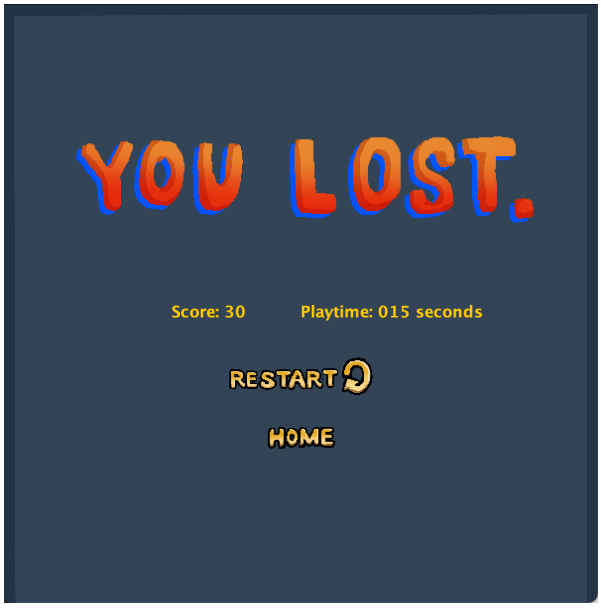
experience in knowing what kind of concerns to plan around for a project like this limited our opportunities and ability to do so. To avoid merge conflicts, we worked out a system where team members would proactively post updates on what they were working on in the group chat channel.This has drastically improved our efficiency and transformed the way we work as a team. In addition, having a team with different schedules and limited time taught us to prioritize. We focused on developing and optimizing the game's basic requirements first, then continued to polish other features to improve the playing experience.

Tutorial

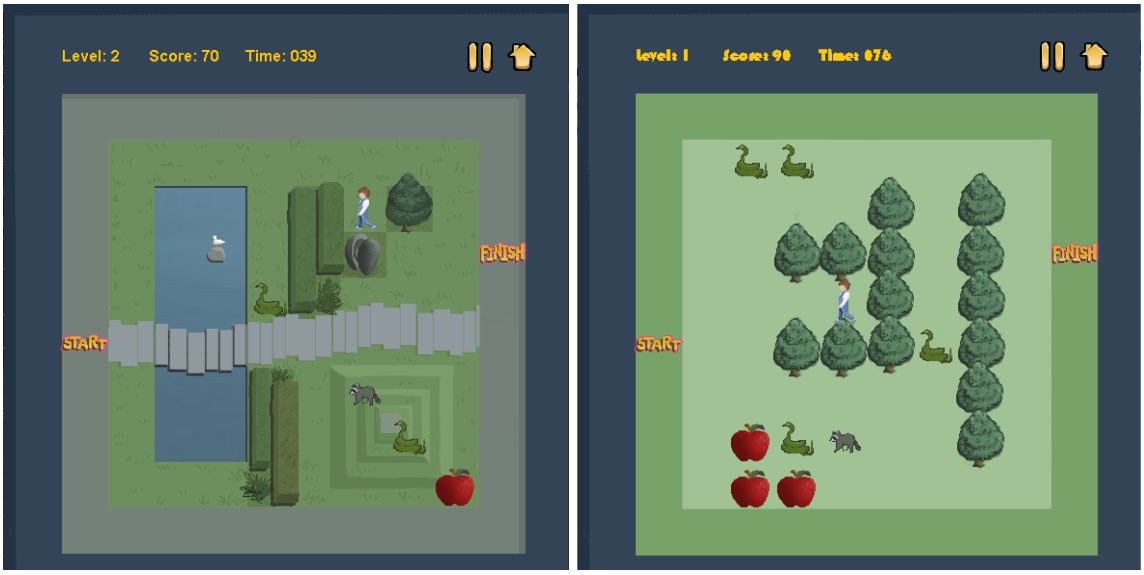

Collect all the apples and reach the finish to win
Move the raccoon by using the arrow keys to choose a direction to move at each tick
The kid will attempt to catch you and will also move at each tick
Neither of you can move past walls or map borders.



For extra points, try to collect the student ID, but move fast or it'll disappear!
You'll also get more points for getting to the finish faster!

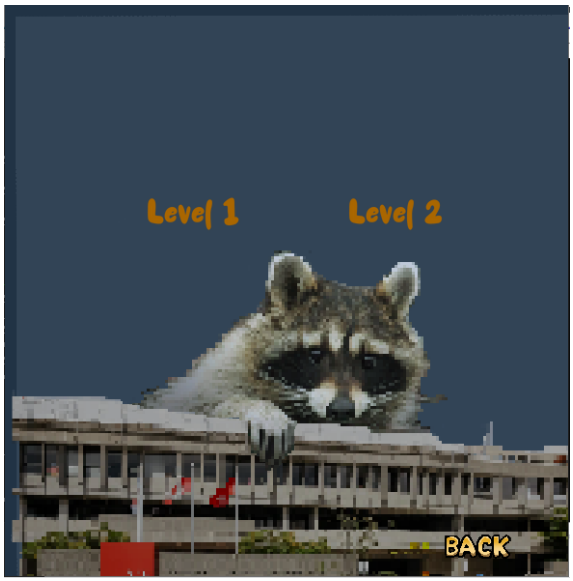


If you step on a snake you will lose points. Lose too many points or get caught by the kid and it's GAME OVER. Don't forget there is a time constraint so you better hurry!



If you’re up for the challenge you can try trapping the kid in the forest!

Need to take a break? Use the buttons in the top right to pause the game or quit back to the main menu



Want to replay a level? You have the option to choose between existing levels.