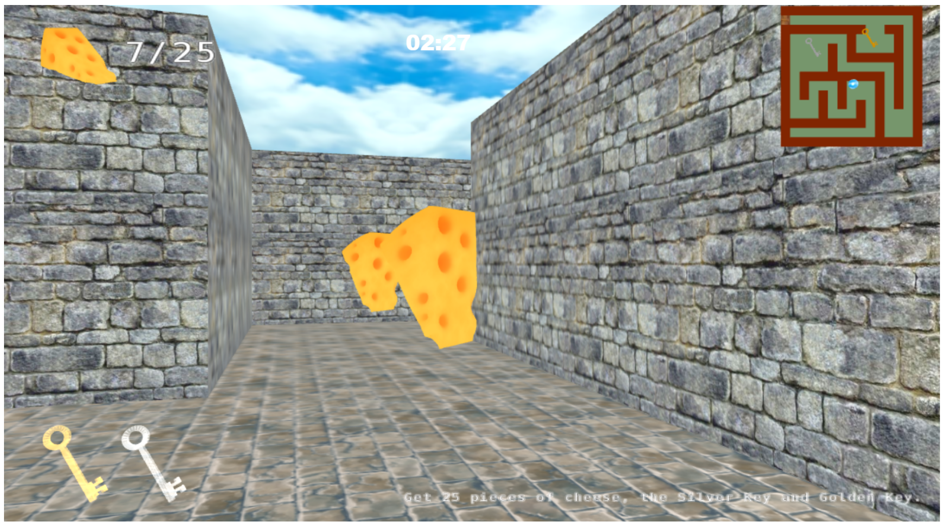
CS 174A Final Report — Group 28

Team name: Hungry Mouse

Name	Email	Student ID
Yuntong Ju	yuntongju@g.ucla.edu	305569504
Yaolan Luo	yaolanluo2002@gmail.com	805973714

Overview

Our project, Hungry Mouse, is an interactive maze game where the player will play as a hungry mouse that navigates the map and collects cheese. The game is first-person-POV, and the player will move around by turning left and right, moving forward and backward. The player needs to collect 25 pieces of cheese in the maze and the golden and silver keys to open the gate, and exit the maze, all within 3 minutes to win the game. The pieces of cheese are randomly generated in the maze. Cheese Count is shown at the top left corner of the screen. The silver and golden key slots are shown at the bottom left corner, and light up if the key is collected. The map of the maze is shown at the top right corner, with a pin indicating the player's location.



Get 25 pieces of cheese, the Silver Key and Golden Key.

Interactivity

The user's perspective is the direction the hungry mouse is going, so the user can rotate the view to change the mouse's direction. Whenever the mouse collides with a piece of cheese, cheese count +1 in the upper left corner. When we collected >= 25 pieces of cheese and the 2 keys (one silver, one gold), the door on the top left corner of the map will open and an exit sign will appear. Otherwise, when time is up, the player loses.

Controls

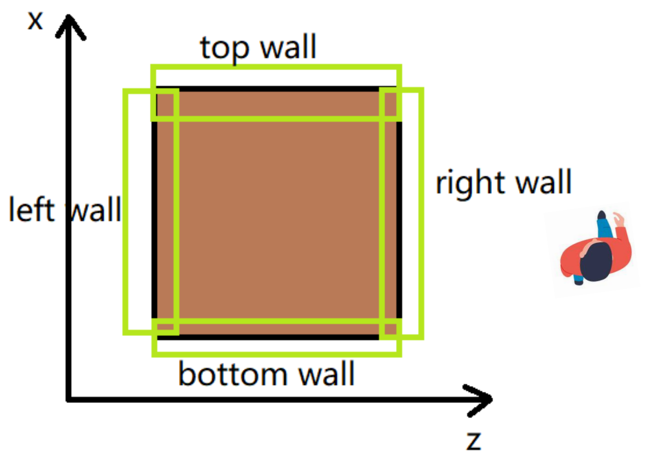
Rotate the view to the left	Arrow left
Rotate the view to the right	Arrow right

Move forward	Arrow up
Move back	Arrow down

Advanced Feature: Collision Detection and Correction

There are two types of collisions used in the game: collision with the wall, and collision with objects (cheese and keys).

Collision with the Wall:



Collision Detection

There are four types of wall collisions, collision with the top wall, bottom wall, left wall, and right wall. As shown in the figure, the brown square is the 10x10 wall block that makes up the maze wall, seen from the top-down perspective. So the top wall is not the top part of the wall, but the part of the wall that is in the greatest x-direction. The z-x plane is the horizontal plane that the player moves in. If the player's x and z coordinates fall in any one of the four rectangles, then a collision with one of the four walls is detected. If the coordinates fall in two adjacent triangles, then a collision with an inner corner (just like the inner corner of a room) is detected.

Collision Handling

A simple algorithm is used to make the player walk along the wall or away from the wall when a collision occurs. For instance, if the player collides with a top wall, the player's movement in the x direction is max(0, dx). dx is the speed in the x-direction. So the player can't go into the wall. The player's dz remains the same. If the player collides into an inner corner, the movements in both the x and z directions are corrected to max(0, dx) and max(0, dz) respectively (or the minimum of 0 and dx, in cases of other types of corner collisions). In all cases, the algorithm prevents the player from going into the wall and covers all possible locations that the player could be in.

Collision with Objects:

The player can collide with the cheese or the two keys that are floating in the maze. This is implemented with the standard AABB collision detection. The shapes of the cheese and the keys are rounded to rectangles with different widths. If the player collides with the objects, they disappear and play a soundtrack. If the player collides with the key, it shows up in the key slot.