Spencer Underhill

Professor Gordon

CSc 165

9 March 2016

Project 2 Report

Compile and Run:

To compile this program, the following command needs to be run from within the root folder of this project. The root folder, unless changes are made, is the folder that is created by unzipping the ZIP file. The command is:

javac a2/.java myGameEngine/*.java*

To run this program, the following command needs to be run from within the root folder of this project. The root folder, unless changes are made, is the folder that is created by unzipping the ZIP file. Windows might require a DirectX disabling command line option. The exact text of this switch is: `-Dsun.java2d.d3d=false` . The command is:

java a2.Starter

In addition to the aforementioned commands, the zip file contains two (2) batch files. Both need to be executed from a command window opened from within the root folder of the project. The first one, compile.bat, will only compile the project. The second, compileAndRun.bat, will compile and then run the program.

Game operations:

This game requires the keyboard and controller to be connected. The program WILL NOT run correctly without the controller connected.

The first player (on the bottom half of the screen) is controlled by the keyboard. The following keys are the controls:

- Esc - quit program
- W – forward
- A – left
- S – backward
- D – right
- Q – rotate avatar left
- E – rotate avatar right
- Z – zoom out
- X – zoom in

- Left arrow – rotate camera counterclockwise, with the camera at 6 o'clock and facing 12 o'clock
- Right arrow – rotate camera clockwise, with the camera at 6 o'clock and facing 12 o'clock

The second player (on the top half of the screen) is controlled by the controller. The following controls are used:

- Left thumbstick
    - Forward – move forward
    - Back – move backward
    - Left – move left
    - Right – move right
- Right thumbstick
    - Left – rotate avatar left
    - Right – rotate avatar right
- Buttons
    - X – rotate camera clockwise, with the camera at 6 o'clock and facing 12 o'clock
    - B – rotate camera counterclockwise, with the camera at 6 o'clock and facing 12 o'clock
    - Left bumper – zoom in
    - Right bumper – zoom out

Both players move their avatars around the game world and intersect with objects in the world to gain points. When a collision occurs, the object is moved into the Doghouse, which will flash white for a small amount of time. The objects will also appear in the Doghouse in miniature.

Scoring:

The players receive a point for each game world object they collide with, except for the Doghouse, the other player, and any objects that have been moved into the Doghouse.

Met Requirements:

- 3rd person camera. This was met by creating a duplication, with modification as required, of the 3pCameraController code provided as an example by Professor Scott Gordon. This is mostly contained in the 3pCamerController.java file. This camera is an orbit camera.

- Multi-player viewports are provided and implemented in the DogCatcher3D.java file. The specific location is in the createPlayers() method.
- The HUDs for each player are implemented in the createPlayerHUDs() method. Both display the Player number, the score (appended after the player number), and the time elapsed.
- A ground plane was added as a SAGE Rectangle, translated, scaled, and rotated to be on the XZ plane. The avatars do not leave this surface.
- The two SceneNode controllers are implemented as, first, RotationController, which handles the rotation of the Pyramind game world objects, which are themselves grouped in a Group. The Rotation applied to the Group, but is specifically unwrapping the Group and performing the same action on each of the Group children. The second, the DoghouseColorController, handles switching the Doghouse color back to its normal color. When an object collection occurs, the Doghouse switches to white. The controller then handles the countdown and then switches it back to standard color.
- The two input controllers are the keyboard and the game controller. Player one uses the keyboard and player two uses the game controller.
- The Group is implemented to include all of the Pyramids. The Player One avatar is a Pyramid, but this is not in the Group. The transform applied is a rotation, which is managed by the aforementioned RotationController.
- The IDisplaySystem is implemented as MyDisplaySystem. It provides the implemented methods required to make the game function (mostly getters and setters) and the IDisplaySystem examples provided by Professor Scott Gordon. The BaseGame implementation overrides initGame(), update(), initSystem(), and render(). InitSystem() is derived from Professor Scott Gordon's provided examples. The render() method sets the camera to the first player's 3pCamerControllerm, calls super.render(), sets the camera to the second player's, then calls super.render() again.
- The game runs in FSEM exclusively. On certain Windows systems, this might require using the command line option to disable DirectX optimizations:
    - `-Dsun.java2d.d3d=false`
- The Doghouse event handling is unchanged from the previous assignment, which responded to SAGE Events.

Unmet Requirements:

I believe all requirements have been met, generally speaking. However, I am aware of two issues that the program has. The first is that I encountered unimplemented functions for collision detection in SAGE (specifically BoundingVolume intersection), so the two camera positions are still used for collision detection. Second, the Left Bumper and B buttons on the controller do not map correctly to their designated functions, though their counterparts do.

Gamepad and lab machine:

This program was tested with an Xbox One Wireless Controller, connected to the computer using a micro-USB cable. No wireless was used. The lab computer tested on was Zork.