Harlan Brawer
Max Zhang
Colton Meskell

Earth Runner
CS 174A Final Report

Introduction

Our project is a game in which the player controls a space-ship-like object (originally designed as a planet) that travels through space, and it has to dodge all the incoming obstacles. This project features many advanced algorithms, such as collision detection, physics based movement, scene transitions, a random number generator, etc, to make the game appear smooth and natural. Below are some explanations of how we implemented those features.

The Animation

For the beginning animation, because we have a theme of a galaxy and we have textures that look like the galaxy, we decided to have some fun and make a star-wars like opening scene. The first feature here is a scene transition. Because the opening scene consists of 2 parts, we need to have different cases, specifically different models, different textures, and different camera positions, according to different animation time. The button “Skip Intro” further complicates this issue, because now with this function, we need to ensure that after this button is pressed, the animation is proceeded to a correct animation time. We also need to ensure that the button can only be effective once, and once the scene has proceeded to the gaming portion, this button will not reset anything.

The actual animation is implemented after the scene transitions. The concept here is trivial: let the texts (squares with text textures) move along the z axis over time, then place the camera at a position such that the text is moving beneath it.

The Earth

The Earth is a subdivision surface object of order 4. It moves in the x and y directions using a normal physics system. It’s acceleration is calculated every frame, the acceleration

updates the velocity, and the velocity updates the Earth's position. The acceleration is calculated based on the movement controller. If one of the i,j,k,l keys is pressed down, the corresponding direction variable is briefly set to 1 then set back to zero. This effectively causes the acceleration to be incremented in that direction when the corresponding key is pressed. To keep the ball from infinitely accelerating, we added a typical velocity based drag force. This makes the movement of the ball feel realistic but still controllable. The ball is textured with a picture of the Earth to give a theme to the game.

The wings of the earth were added to make it feel more like a ship and to give the player the idea that they can steer it. The wings were created using triangle base shapes that are transformed based on Earth's position and scaled to look like wings.

The Grid

The grid is made up of an array of booleans. Each display cycle, the array is iterated over and if the spot in the grid is a 1, then a cube is drawn to the screen at the corresponding location in the world. The grid starts at a location far in front of the camera and moves towards the Earth and the camera. When the grid moves behind the camera it is regenerated back at its starting position. When the grid is regenerated, the booleans in the array are randomly reset. To do this the array is iterated over and for each index, a random number is generated between 0 and 1 and if the number is above some threshold then the index is set to 1 and otherwise 0.

Collision Detection

The only collision detection is between a sphere and cube of the grid. To check collision between a cube and a sphere, the nearest x, y, and z of the cube to the sphere are calculated. These are then used in a distance formula to determine if the cube and sphere are overlapped. Collision is checked for each cube in the grid only when the grid is close enough to the ball to be able to collide. This prevents needless collision detection that could cause the game to lag.

The Camera

The camera moves based on the ball transform plus a translation such that it constantly stays behind the Earth, looking at the grid. To give the motion of the earth more of a feeling of being in a ship, we added an extra camera rotation based on the left-right velocity of Earth. This causes the camera to rotate as you move right to left, giving you the feeling of a ship tilting as you move it.

Random Textures and Shades

The textures on the objects change at random due to a random function that chooses a texture or color at random and shades over that level. This is done with an array of materials using Math.random()*array size and flooring that number to give a completely random cube texture each time. The background was drawn onto a flat square set behind the scene.