Jeel Patel

[jp11223@georgiasouthern.edu](mailto:jp11223@georgiasouthern.edu)

Jonathan Zheng

[sz00972@georgiasouthern.edu](mailto:sz00972@georgiasouthern.edu)

Click3D

**Introduction**

Click3D is a game designed using Java3D API. The premise of the game is to click on as many spheres as possible to accumulate points before the timer runs out. The game runs on a 60-second timer which starts after clicking the start button on the main menu. The main menu also displays the current high score of the player and updates each time a player achieves a personal best. Once a player clicks on the start button, the game begins with a sphere object rotating around the game screen at a random location. If a player clicks on the object, it will shift to a new random location and the player’s score will increase by 1. Alternatively, if a player fails to click on the object within 2 seconds of the object appearing, their score will be deducted by 1. This will continue for the full duration of 60 seconds of game time. However, the game progressively gets more difficult the higher the player scores. With a score of 0, the scaling is set to 0.25. However, every 10 points the scaling decreases by 0.5 until the scaling reaches 0.05. At which point it will stay for the rest of the game. Once the timer reaches 0, the game will end, and the player will be redirected to the main menu. The player can then begin a new game if they wish to beat their high score.

**Game Elements and UI**

Click3D is mainly designed using Java3D. However, it uses other java libraries to accomplish various functions, such as java.util.Timer to keep track of game time. Click3D extends Applet and implements MouseListener and ActionListener to read mouse inputs. Upon launching the application, the first thing a player sees is the main screen. The main screen has a start button and a high score text. The start button launches the game screen, and the high score text displays the highest score a player has achieved. The high score text is read from a highscore.txt file, which is updated each time a player achieves a new high score. The start button and the text area are made using JButton and JTextArea, respectively. They are housed within a panel with a GridLayout. The panel is then added to the mainframe of the application. Once the start button is clicked, the startGame() method is called which creates a canvas3D with a mouse listener. A score and time text areas are all added in the top left corner of the application. They are both created using JTextArea and added to a panel object. A new Timer object is also created to keep track of current game time. It uses a TimerTask, which creates a new thread and counts down from 60 to 0. This thread also tracks elapsed time which calls the newScene() method every 2 seconds and decreases the player score by 1. Once the timer hits 0, the game is ended and the stopGame() method is called. The startGame() method also creates a branch group with capabilities that allows children read, children write, children extend, and pickable write. A new pick canvas is also created to allow mouse clicks to interact with 3D objects on the screen. The branch group is then added to the simple universe, which creates the play area for the game. The stopGame() method removes all the elements from the application and calls the startNewGameMainMenu() method, which displays the main menu. The newScene() method removes the current branch group from the simple universe and recreates a new branch group with a pick canvas. The newly created branch group is added to the simple universe.

**Mouse Click**

Canvas3D is created with startGame() method and has a mouse listener. Each time a mouse click occurs, the mouseClicked(MouseEvent e) method is called. This method gets the mouse location using the pick canvas and finds the closest shape to the click. If a shape is found, the newScene() method is called to create a new shape, and the play score is updated by 1. Additionally, a check is performed to see if the current score is higher than the highest score of the player. If it is, updateFile() method is called, which creates a highscore.txt file using FileWriter class and updates the file with the new high score.

**Scene Graph**

Click3D uses SimpleUniverse which provides a framework for 3D rendering, and Canvas3D which displays 3D rendering. A scene graph is created for this application using the createSceneGraph() method which returns a BranchGroup. In this method, the BranchGroup object that is created at the beginning acts as the root, which holds all the elements. It also creates a new TransformGroup() object, with transform write and transform read capabilities. It is then added to the root branch group. After creating branch group and transform group objects, generateSphere() method is called, which creates a 3D sphere object using the Sphere class. The sphere object is attached to an instantiation of the Transform3D class, which sets the scale of the sphere according to the player’s current score. It also sets a random translation using a randomly generated Vector3f. Once the sphere object is created, several classes are used to create the lighting and the background. Firstly, a BoundingSphere object is instantiated, which will define the range of the lighting. The Background class is used to set the background color. AmbientLight, DirectionalLight, PointLight, and SpotLight classes are used to generate the various lighting features that the sphere object has. Each time a new sphere object is created, a random lighting effect is given to the object. Additionally, a RotationInterpolator object is used to rotate the sphere around the canvas. Finally, the root branch group is returned by the method.