SFWR ENG 3GC3

SphereQuest

Project Documentation

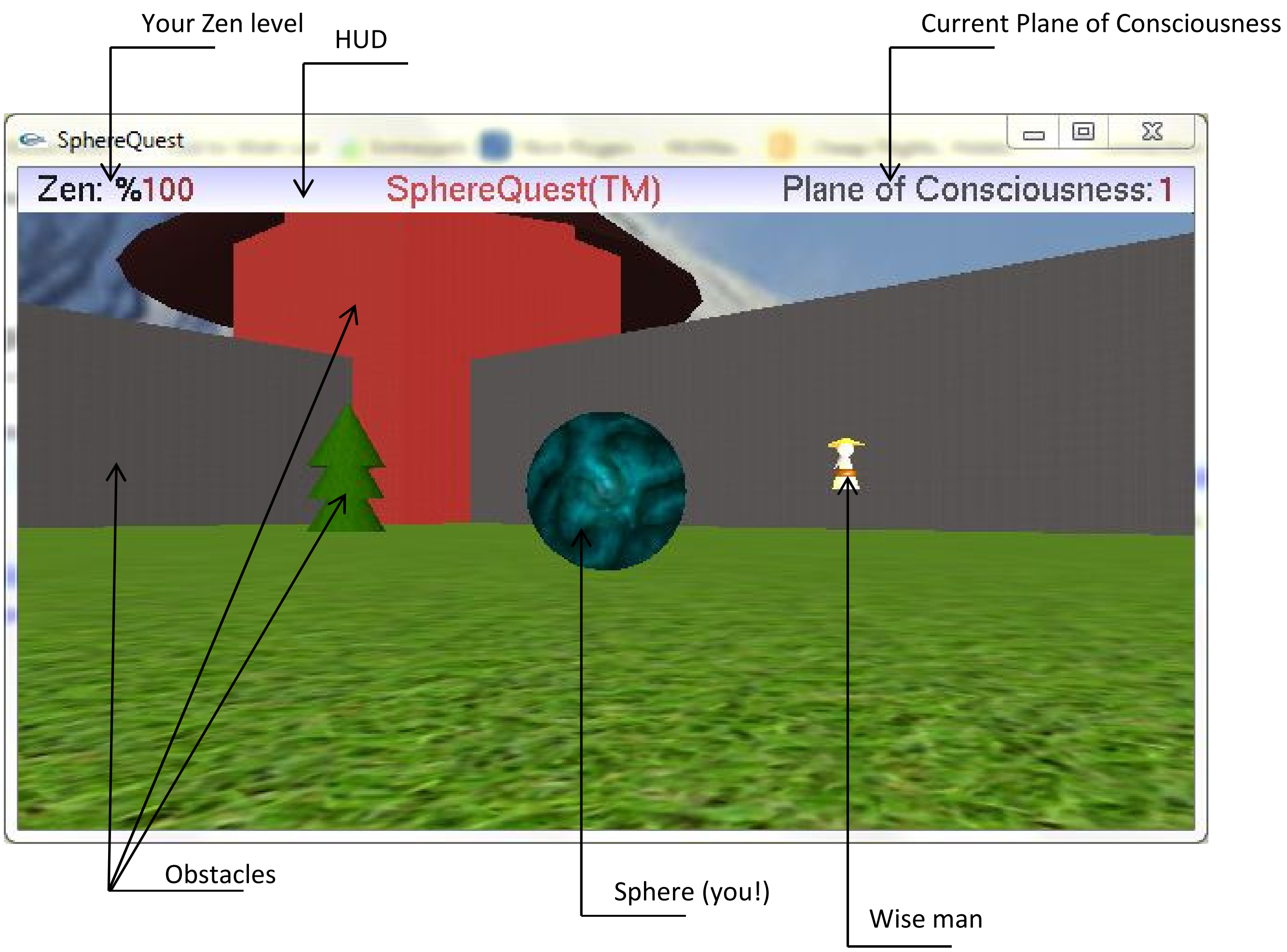
Jason Costabile, Ben Kybartas, & Kaitlin Smith

4/5/2010

Introduction

Welcome to SphereQuest! SphereQuest is a game designed to bring you inner peace by bringing you on a personal journey through your inner self. Find the wise men throughout the mazes and answer their riddles to ascend to a higher plane of consciousness!

User Manual



Objective

The objective of SphereQuest is to reach the sixth Plane of Consciousness, where your spiritual journey of self-realization will come to an end. Before you can ascend to the next Plane, you must locate the wise man and successfully answer his riddle.

Installation

To install SphereQuest, simply unzip the provided archive into your preferred directory. Run the executable file to play the game.

To uninstall, delete the SphereQuest folder where the unzipped files were placed.

Choosing Difficulty

When SphereQuest is started, you will be presented with a splash screen and three difficulty levels to choose from. A harder difficulty level causes you to lose a higher amount of zen when a riddle is answered incorrectly. Choose a difficulty level using the number keys on your keyboard.

Movement

To navigate the sphere through each Plane, simply press the arrow keys on your keyboard. The sphere will glide in the direction of the arrow key currently pressed.

The sphere is not allowed to simply move where it pleases. Throughout the maze, you will find obstacles which you cannot pass though – walls, trees, and temples. You may not move travel through the wise men, but it is easy enough to move around them.

Riddles

Once you have successfully located a wise man, approach him to be asked a riddle, along with a choice of three answers. Search your soul and think carefully about the answer. Once you have made your decision, press the function key that corresponds to your chosen answer – each answer is labelled with the key to press.

If you have chosen the correct answer, the sphere will move to the next Plane of Consciousness. This can be confirmed by reading the Plane of Consciousness indicator in the top-right corner of the HUD (heads up display).

If you have chosen an incorrect answer, you will lose a portion of your Zen. You may monitor your Zen level by reading the Zen meter in the top-left corner of the HUD. Approach the wise man again for another opportunity to answer his riddle.

Winning and Losing

Winning SphereQuest occurs when you have successfully reached the sixth Plane of Consciousness before your Zen reaches zero.

Losing SphereQuest occurs when you have answered too many riddles incorrectly, causing your Zen to reach zero. If this happens, you will be presented with the option to try again. If you decide to try again, you will appear back on the first Plane of Consciousness, ready to retry your spiritual journey.

Saving and Loading

If you need to take a break from your journey, you may save your current progress. Right-click anywhere within the game window to be presented with the Save/Load Menu. Choose a slot to save in and a save file will be created for you. To load a game you've saved, right-click again and choose the slot to load in which you saved your game. When saving a game, take care to not save over another journey, as it is not possible to recover a saved-over file.

Cheats and Developers' Shortcuts

Several shortcuts were implemented to aid in the development of SphereQuest, and can be used as cheats if desired.

To display grid lines, press the J key. Press it again to remove the grid lines.

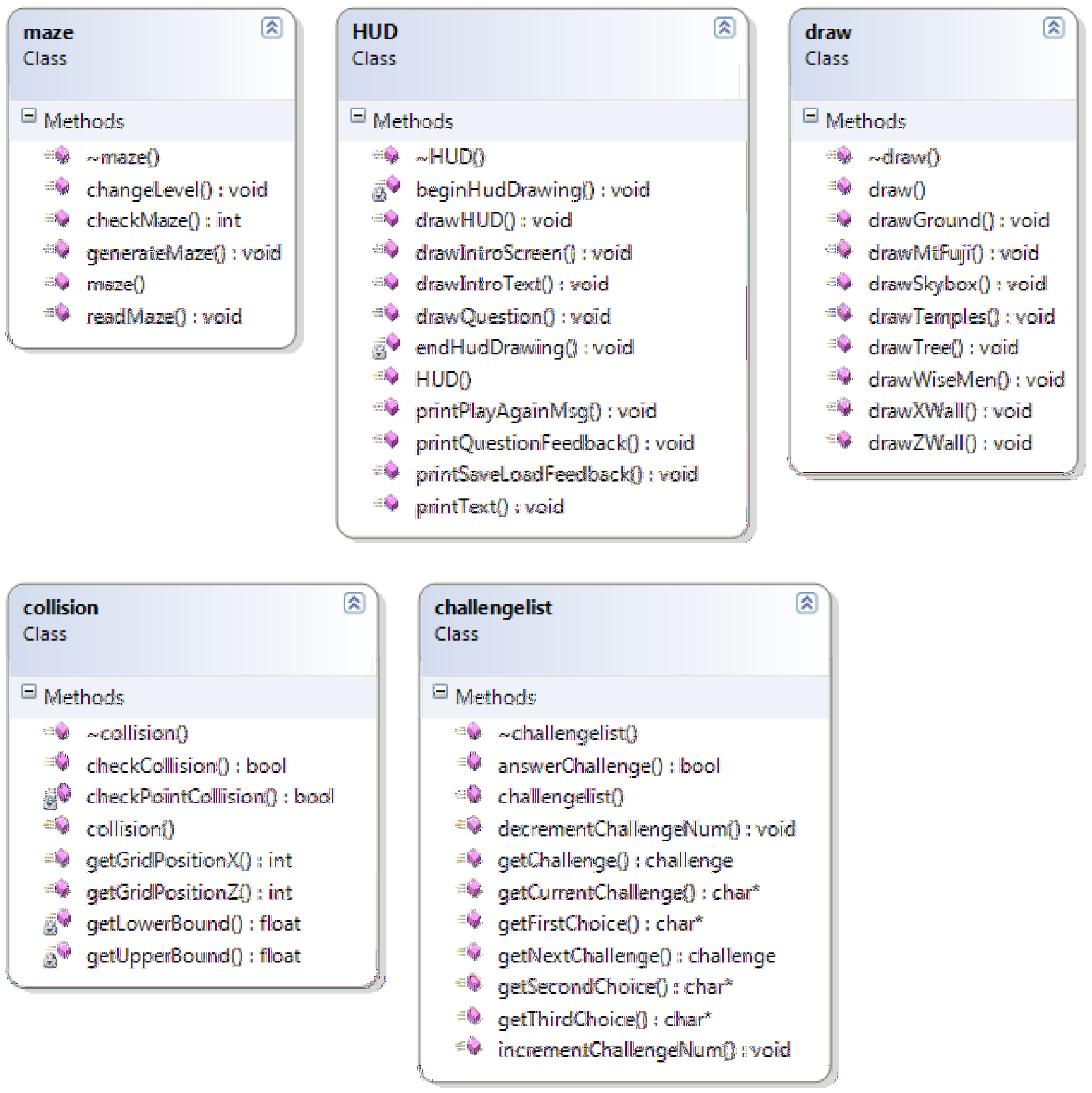
To skip the current Plane of Consciousness, press F4. To automatically win, press F5. Pressing F6 will cause you to automatically lose.

Code Documentation

Please make sure to browse through the code as there are many comments documenting the purpose of individual code chunks.

Class Breakdown

A quick breakdown of the classes in the SphereQuest project is given in the class diagram below.



The SphereQuest project does not use class inheritance (each class is independent), so no connections are drawn between the classes. Public methods are represented by a purple brick, which private methods are represented by a small lock beside the purple brick. Every class contains a constructor and destructor method.

The maze Class

The maze class contains all code for creating the maze in which the user navigates through.

The checkMaze Method

This method takes integers representing a row and column of the maze and returns an integer representing the item that is located in that grid location. The table below summarizes the grid items:

Integer	Grid Item
1	A wall placed on the x-axis.
2	A wall placed on the z-axis.
3	A temple.
4	A wise man.
5	A tree.

The changeLevel Method

This void method takes an integer representing a level number and loads the corresponding maze file from the provided maze folder. The loading of the file is handled by the readMaze method described below. This method also modifies a global level variable and sets it to the changed level.

The readMaze Method

This method takes a string representing a file name and loads that requested file to read the maze from. Maze files are located in a 'mazes' folder and the maze data is stored in text file. Please take the time to examine the maze files and note their simple structure.

If the requested maze file is loaded successfully, the method will read the file, character by character, and store the current character in the mazeLayout array declared globally in the maze class. If the method is unable to load the file, an error is printed to the console.

The HUD Class

The HUD class takes care of drawing the elements of the heads up display.

The printText Method

This method takes two floating point numbers representing screen coordinates, a character array representing the text to be printed, and three floating point number representing an RGB value. This method then applies the given colour, positions the text on the screen, and then prints the character array using glutBitmapCharacter.

The drawIntroText Method

This method uses the printText method above to print the text which appears when the game first starts. The beginHudDrawing method is also used, which is described below.

The drawQuestion Method

This method takes several character arrays as inputs which represent the question to be printed as well as the three choices for answers. The printText method is used once again to display the text. The beginHudDrawing and endHudDrawing methods are also used, which are described below.

The printPlayAgainMsg Method

This method accepts a Boolean value representing whether or not the player has won the game. If the value is true, this method uses the printText method to print a message indicating that the player is victorious. If the value is false, this method uses the printText method to print a message indicating that the player has lost. In both cases a line of text is also displayed prompting the user whether or not he would like to play again. The beginHudDrawing and endHudDrawing methods are also used, which are described below.

The printSaveLoadFeedback Method

This method accepts two Boolean values, one representing a save or load took place, and whether or not the action was successful. Depending on the values of the Booleans, if statements choose which messages to print using the printText method. The beginHudDrawing and endHudDrawing methods are also used, which are described below.

The printQuestionFeedback Method

This method takes a Boolean value which represents a correct or incorrect answer to a riddle. If the answer is correct, the method will print a success message to the screen. Otherwise, a failure message will be printed.

The drawHUD Method

This method's responsibility is to display the HUD. It uses the global variables zen and level to get information about the player's current state. The method converts these variables from integers to strings in order to display the information, using the printText method described above. It also draws a grey rectangle to display the text on top of. The beginHudDrawing and endHudDrawing methods are also used, which are described below.

The drawIntroScreen Method

This method simply draws a quad where the splash image will be placed. The placement of the image occurs elsewhere.

The beginHudDrawing Method

This method simply sets up the matrices that are used for drawing the 2D HUD. It must be called within each method which modifies parts of the HUD.

The endHudDrwaing Method

This method simply pops the matrices which were used to draw elements of the HUD. It must be called after any code which is used to modify the HUD.

The draw Class

This class is contains the methods which are used to draw the elements of the SphereQuest world. Each method simply contains the OpenGL commands needed to draw the element. Elements with animation have the animation implemented in the drawing methods. Please examine the code for comments detailing each part of the drawing process for each element.

The drawTemples Method

This method draws the red temples.

The drawXWall Method

This method draws a wall moving in the x-direction.

The drawZWall Method

This method draws a wall moving in the z-direction.

The drawWiseMen Method

This method draws the wise men.

The drawTree Method

This method draws the trees.

The drawGround Method

This method draws the ground.

The drawSkybox Method

This method draws the skybox surrounding the game world.

The drawMtFuji Method

This method was meant to draw Mount Fuji using primitives, but it is not currently used in the game.

The collision Class

This class is responsible for detecting and dealing with collisions between the sphere and the obstacles in the game world.

The collision Method

This method takes a pointer to a maze as an input and stores it in the global levelMaze variable for later use.

The getGridPositionX Method

This method takes an x-coordinate and returns an integer corresponding to what is drawn in that grid location. See the table above for information on which integer corresponds to which element.

The getGridPositionZ Method

This method takes a z-coordinate and returns an integer corresponding to what is drawn in that grid location. See the table above for information on which integer corresponds to which element.

The getUpperBound Method

This method accepts two floats representing the width of an obstacle and the tolerance amount. This “draws” the obstacle’s bounding box’s upper limit.

The getLowerBound Method

This method accepts two floats representing the width of an obstacle and the tolerance amount. This “draws” the obstacle’s bounding box’s lower limit.

The checkPointCollision Method

This method accepts an x and z-coordinate and checks if this point would be colliding with an obstacle in the game world. The method returns true if a collision has occurred and false otherwise. A case-switch statement determines exactly which obstacle is being collided with. Since each obstacle is a different size, an if-statement within each case-switch checks whether the point is colliding with that exact obstacle.

The checkCollision Method

This method accepts three coordinates and an object radius, each of which are represented by floats. This method checks if the point at the given coordinates will collide with an obstacle’s bounding box.

The challengelist Class

This class handles the list of riddles presented to the player, as well as the progression through the riddles.

The challengeList Method

This method, which is also the constructor, contains the hard-coded array of riddles and their answers, as well as a variable which keeps track of which riddle the user is currently being asked.

The setChallengeNum Method

This method takes an integer as an input and sets the current riddle to this integer.

The getChallengeNum Method

This method returns the current riddle number.

The incrementChallengeNum Method

This method increases the riddle number, which allows the next riddle to be displayed.

The decrementChallengeNum Method

This method decreases the riddle number.

The getCurrentChallenge Method

This method returns the current riddle as a character array.

The getFirstChoice Method

This method returns the first answer choice for the given riddle.

The getSecondChoice Method

This method returns the second answer choice for the given riddle.

The getThirdChoice Method

This method returns the third answer choice for the given riddle.

The answerChallenge Method

This method takes an integer, representing the user's choice, and returns true when it is the correct answer to the riddle, and false when it is the incorrect answer.

The GameBase

The GameBase is the main file of the game. Many methods are declared in this file which could not be moved to a class – this was usually because creating a class with these methods would complicate the program. Any important methods are described below, otherwise see the comments in the code for information on all methods.

The loadTexture Method

This method takes a texture image file and its width and height, and loads the texture for later use.

The moveCamera Method

This method takes x, y, and z-coordinates, which then moves the camera to this position. This method causes the camera to always look at the sphere.

The changeSize Method

This method takes in two integers which represent the window width and height for the program. This method allows the user to attempt to adjust the window size, but it will always snap back to the intended size. This method has been coded this way to allow future changes to the window size if it is desired.

The drawSphere Method

This method is responsible for drawing the sphere. This method was placed in GameBase because so many of the variables used in the method were used in GameBase that it was difficult to place this method in the draw class.

The drawGrid Method

This method is used to draw white grid lines on the map. It was used to debugging as there were some issues with collisions at the beginning. These lines helped the developers visualize where the collisions were supposed to occur.

Look into your Heart

Dedicated to Mr. Oscar Basurto Carbonell