Computer Graphics Project

Recreating Nether Earth

Team 2

Jonathan Bergeron

Jeff How

Robert Jakubowicz

Stéfanie Lavoie

Addison Rodomista

Table of Contents

[Introduction 3](#_Toc319490087)

[Project Design 3](#_Toc319490088)

[Logic 3](#_Toc319490089)

[Model 3](#_Toc319490090)

[Buildings 4](#_Toc319490091)

[Helper 4](#_Toc319490092)

[Material 4](#_Toc319490093)

[Player 4](#_Toc319490094)

[Robot 4](#_Toc319490095)

[Skybox 5](#_Toc319490096)

[Static 5](#_Toc319490097)

[Texture 5](#_Toc319490098)

[Helper 5](#_Toc319490099)

[User Manual 6](#_Toc319490100)

[Appendix 7](#_Toc319490101)

# Introduction

The purpose of this document is to provide the reader with an in-depth understanding of our version of Nether Earth. There are two main sections in this guide. The first section, Project Design, will discuss the overall class structure and the logic behind each of our classes. The second section, User Manual, will accommodate the user by providing keyboard commands.

# Project Design

Refer to Exhibit 1 in Appendix for Class Diagram and Exhibit 2 in Appendix for model screenshots.

## Logic

**LevelRenderer:** Class containing the layout of the level and the logic to draw the different tiles.

**Base:** Class to render the main Base on the map

**Camera:** Abstract class used for camera orientation and movement.

**CirclingCamera:** Bird eye view camera. Rotates around the scene. A spotlight can also be toggled on and off to where the camera is looking at.

**CommanderCamera:** The strategy game type camera. Player can also zoom in and out and rotate it around the look-at point.

**FreeLookCamera:** Mouse look camera view.

**Game:**Where game logic is computed.

**LightCamera:** Creates a new LightCamera and is used to adjust the camera position according to specified light positions on the map

**Player:**Where information on a player is held.

**PlayerInput:**Where player-related commands are parsed and executed.

**Robot:** Encapsulates and manipulates robot-related, such as components, coordinates and orientation.

**RobotCamera:** Child class of Camera. Used for Robot View.

**SpotLight**: Sets the standard spotlight components used for all instances of spotlights on the field

## Model

Please refer to Exhibit 2 in Appendix for screenshots of the models.

**Model: Base class that all models must inherit and define the render() method.**

### Buildings

**BaseModel: Using the GeoHelper functions, renders the main Base Model components**

**FactoryModel: Using the GeoHelper functions, renders the Factory components.**

### Helper

**GeoHelper: Contains various geometric functions used to help with other models. Also contains a function to calculate normals.**

### Material

**Material:** Base class that every material inherits. It controls the reflection, shininess and the diffuse properties.

**DefaultMaterial:** The default material that is applied to all models none is provided. It is a normal material with no reflection and no shininess.

**MetalMaterial:** Gives a metallic look by being very shinny.

**OrganicMaterial:** Applies to object such as grass by giving a higher tint of green.

**RockMaterial:** Applies to object such as rock and walls by giving a higher tint of brown.

### Player

**AntennaModel:** Defines the vertices for the small cylindrical object that is on top of the player.

**PlayerModel:** Defines the vertices for the player model.

### Robot

**AntiGravModel: Model that represents the Anti-Gravity component.**

**BipodModel: Model that represents Bipod component.**

**CannonModel: Model that represents the Cannon component.**

**ElectronicsModel: Model that represents the Electronics component. Created using various ‘glu’ quadrics.**

**HeadlightModel: Model that represents the robot headlight.**

**MissileLauncherModel: Model that represents the Missile Launcher component.**

**NuclearModel: Model representing Nuclear component. Created using gluCylinder.**

**PhaserModel: Model that represents the Phaser component.**

**TracksModel: Model that represents the Tracks component.**

### Skybox

**CubicSkybox:** Model for skybox.

**SphericSkybox:** Model for sky-sphere.

### Static

**FenceModel:** Model for the fence that is around the map.

**GrassModel:** The model that represents the floor.

**HalfHollowBlockModel: Rendered using GL\_TRIANGLES.**

**HalfPlainBlockModel: Rendered using GL\_TRIANGLES.**

**HillsModel: Rendered using GL\_TRIANGLES.**

**HollowBlockModel: Rendered using GL\_TRIANGLES.**

**LightRubbleModel: Model for rubble. Rendered using gluSpheres.**

**MountainModel:** Model containing 3 peaks to represent mountains.

**PitModel:** Model representing pit terrain. Contains scaled blocks for foundation and cylinders and disks for smoothing.

**PlainBlockModel: Rendered using GL\_TRIANGLES.**

**TeamNumberModel:** Model of our team number. It is often reused on other models.

### Texture

**Imageloader:** Used to convert .bmp files into Image objects.

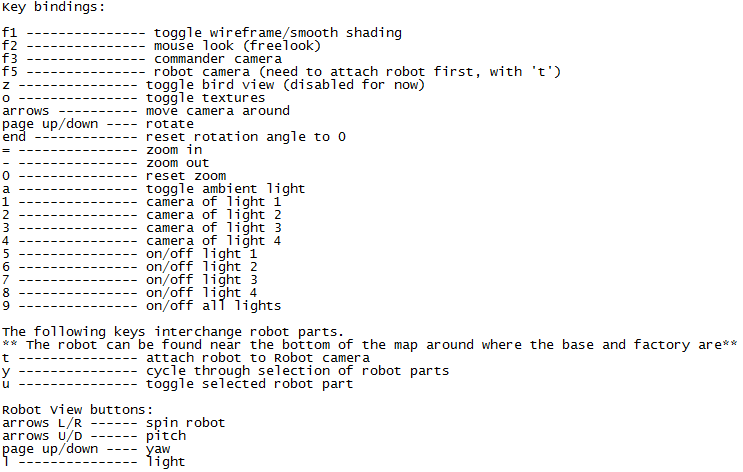
**TextureManager:** Singleton class called by models to find textures. Uses a map to make textures easily retrievable.

## Helper

**AntTweakHelper:** Helper class to help setup AntTweak and bind variables from the game.

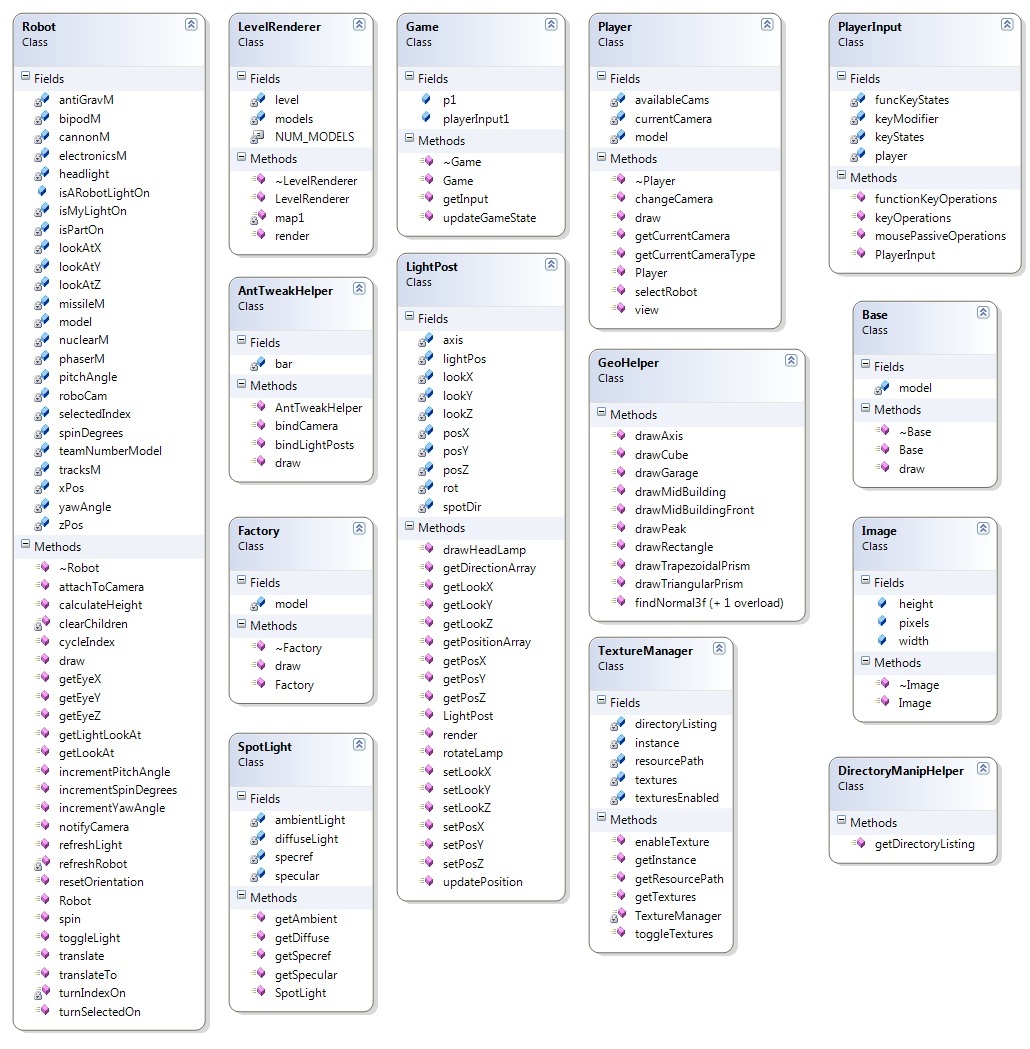
**DirectoryManipHelper:** Helper class used by TextureManager to retrieve the file names for the texture files.

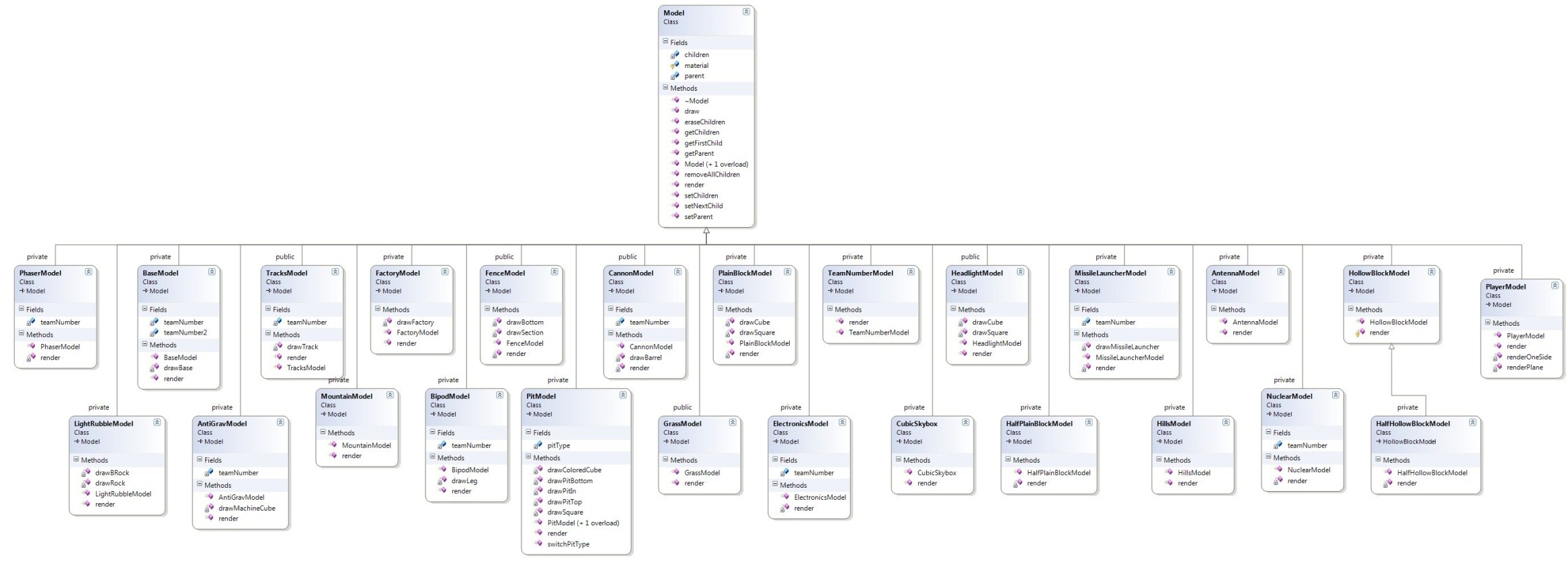
# User Manual



# Appendix

Exhibit 1: Class Diagrams





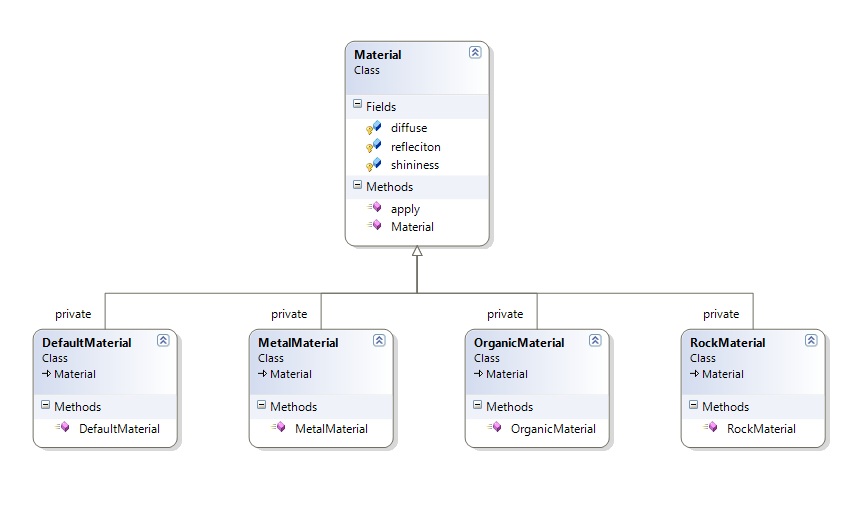
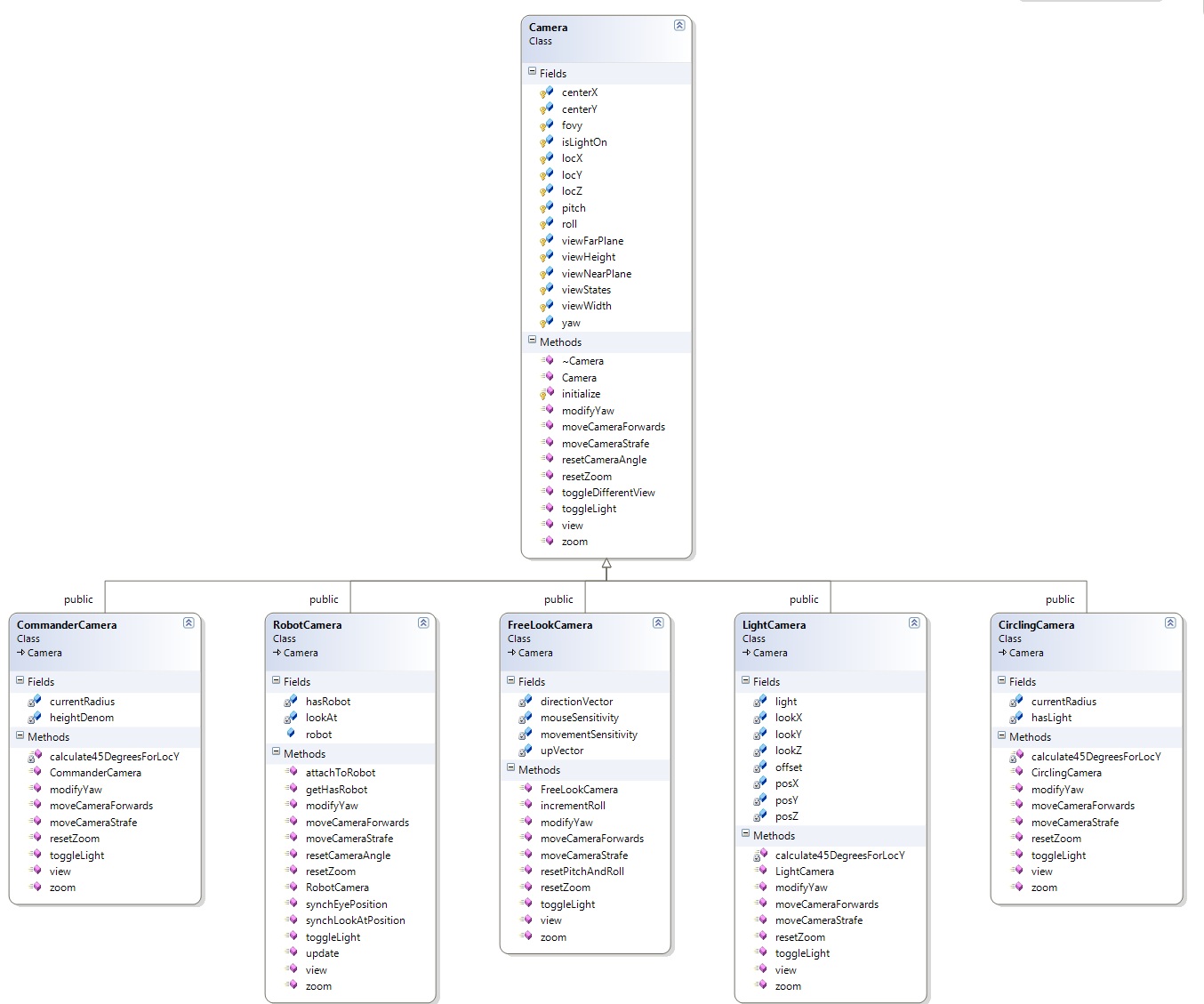


Exhibit 2: Screenshots

