Michael O’Keefe

3D Computer Graphics

Final Project Overview

**Description**

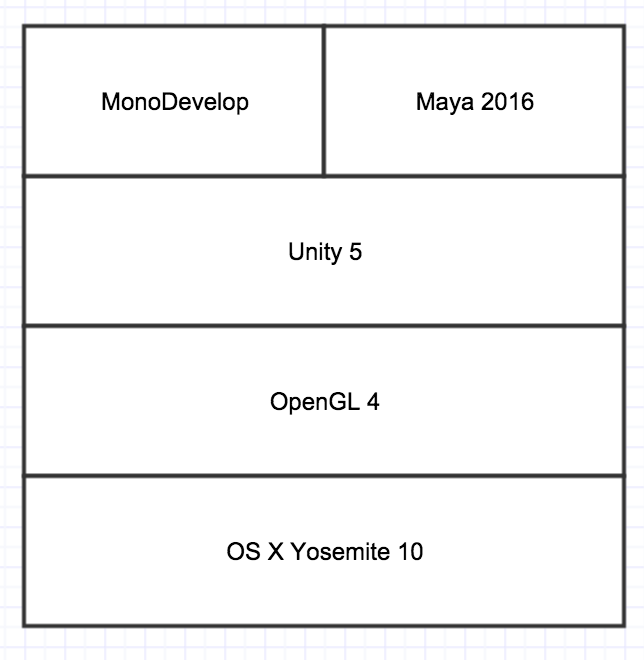
I have developed a “little planet” that exists within an outer space environment (the main object is the planet). The planet contains grass, trees, flowers, a pond, butterflies, fish, a house, and background music. The light source comes from a distant “sun” that looks to be constantly moving around the planet and casting shadows. The outer space background is static, but also appears to be moving around the planet as the sun does. The user is able to to move the camera around the planet by using the W, A, S, & D keys. W for moving forward, S for moving backward, A for moving left, and D for moving right. The user may also use the Z and X keys to zoom in and out. Z for zooming in, and X for zooming out. The camera’s center is always fixed on the center of the planet (the core), and the zoom in and out functionality is set to certain limits so that the user cannot zoom in too close or zoom out too far.

Little Planet was never intended to be a game, but rather a unique environment that is aesthetically pleasing and relaxing for the user to explore. There is no main objective and the cartoon art style adds to the user’s experience. The main influences for Little Planet are from Super Mario Galaxy, and Animal Crossing (both Nintendo games). Please enjoy.

**Final Design**

Little Planet was created on the Mac OS and was put together using Unity 5 and OpenGL 4. Most of the modeling was created myself in Autodesk Maya 2016, and all scripting was done by myself in MonoDevelop C#. Diagram 1: Development View, describes how the software is decomposed for development.

Diagram 1: Development View



The object hierarchy for Little Planet in Unity follows a straight forward structure. Every object that exists on the Planet is a child of the Planet. The Camera is also a child of the Planet so that the camera follows the Planet as it rotates (making the static background appear to be moving).

The Sun is its own separate entity from the planet since its position is static (although it appears to be rotating since the camera is). The object hierarchy is shown in Picture 1: Object Hierarchy.

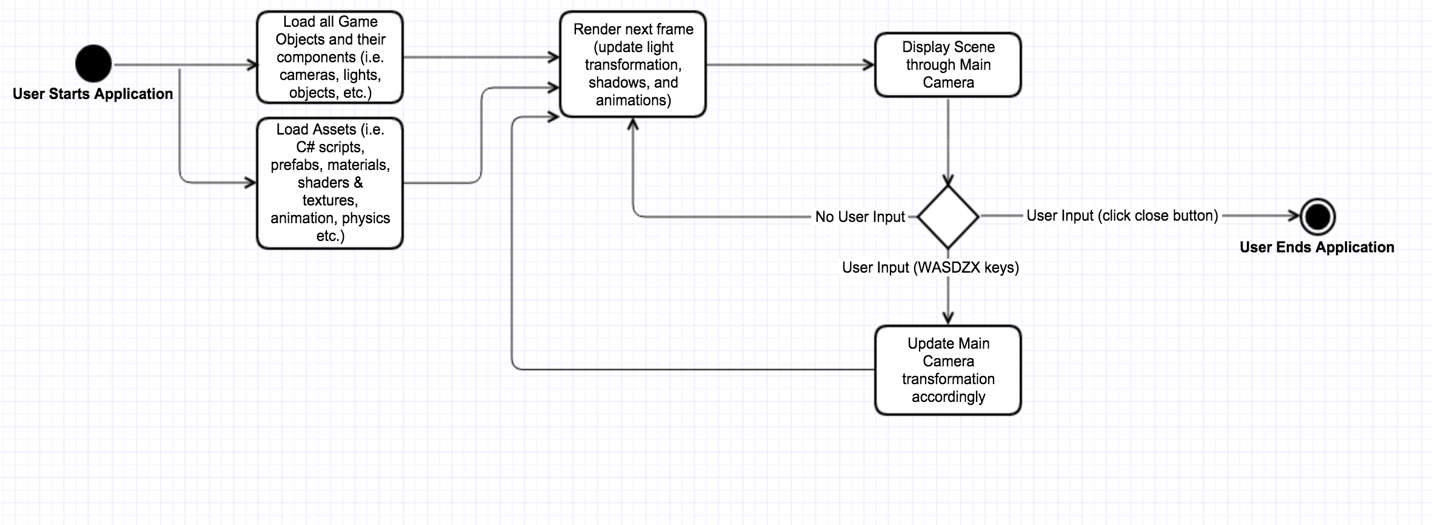
Picture 1: Object Hierarchy



Since Little Planet is not a game, the run time process of the project is not a complicated one. The bare bones idea of it is the scene must be rendered and updated at each frame (update the light, transformations, animations, and shadows), and user input must be updated accordingly.

Diagram 2: Process View, describes in more detail how, at run time, the project is composed of interacting processes.

Diagram 2: Process View



**Features**

Before discussing any features, please note that I modeled, textured, animated, and shaded all objects except for the dark butterflies (the 8 butterflies that float around the patched of flowers), the flowers, and the fish.

In order to convey how my project has changed since the initial design proposal a few weeks ago, I will list the functional and non functional requirements that were originally planned, but I will make comments underneath each requirement if anything has changed. Most of the changes are found in the required realness of objects. For example, at first I wanted the grass to look real, but now I want it took look cartoon-like in order to match the style of the project. Note that all functional and non-functional requirements have been met.

List 1: Functional Requirements, and List 2: Non-functional Requirements, list the required features of my final project.

List 1: Functional Requirements

1. A “little” sphere planet that has a radius of 10 (based on Unity’s units) and has a center of gravity coming from the core of the planet. Must have surface detection so that objects do not go through the surface due to the pull of gravity.
   1. The planet does have surface detection but it is not needed since all objects have been manually placed in the correct position around the planet.
2. A grass-like terrain the covers all of the planet that is not covered by other objects. The grass must look like real world grass in texture, transparency, color, and reflection.
   1. The grass is not as realistic as possible, but that is on purpose. The realism takes away from the cartoon art style that this project has.
3. At least 3 patches of flowers exist on the planet (includes 5 or more flowers per patch). Flowers can be any color, and shape. Their transparency, and texture must resemble real world flowers. They must have collision detection so they can be on the planets surface and cast shadows.
   1. The collision detection exists but is not needed.
4. Have at least 5 trees that each have their own unique shape and colors. Their transparency, and texture must resemble real world trees. They must have collision detection so they can be on the planets surface, be detected by butterflies, and cast shadows.
   1. The textures are more cartoon-like than “real world”.
5. There will be a pond that is at least 3 Unity units deep and at least 5 Unity units in length (from at least one end, to another). The water must resemble real world water in texture, transparency, color, reflection, and physics.
   1. The water is more cartoon-like than real world.
6. There must be at least 3 butterflies, each with different coloring. Each butterfly will fly close to the surface of the planet (at most 3 Unity units above). Each butterfly must have a flying animation that closely resembles a real life butterfly. The butterfly must resemble a real life butterfly in texture, transparency, and reflection. Surface detection is needed to avoid collisions with other objects and shadow casting.
   1. The butterfly is more cartoon-like than “real life”.
7. A distant light source is needed to represent the sun. It must be bright enough to aluminate the planet where the light makes contact. The light must orbit around the planet at a constant speed, casting new shadows as time goes on.
8. The user must be able to traverse the planet, using a first person camera, by clicking and dragging the mouse over the planet. The cameras center focus is always fixed on the center of the planet. The camera is always a fixed position from the center of the planet (no zooming in or out). The camera should be fixed at a distant position away from the center of the plant where the user is able to see the outline of the planet at any time, and also see the starry background.
   1. Added zoom in and out functionality.
9. Mouse controls must be intuitive. A mouse drag that starts at the top of the screen and moves downwards, should move the camera around towards the top of the planet. All other mouse drags should follow the same directional logic.
   1. Controls are now W, A, S, & D for movement around the planet and Z & X for zooming in and out.
10. The background, surrounding the entire scene, should be a static (non-changing) universe filled with distant stars. Should be dark and beautiful.

List 2: Non-Functional Requirements

1. Must run at at least 30 frames per seconds.
2. The application must load within 5 seconds or less.

Now that we know what my functional and non-functional requirements were, lets look at the features I delivered. List 3: Implemented Features, and List 4: Implemented Bonus Features, describe all of my features.

List 3: Implemented Features

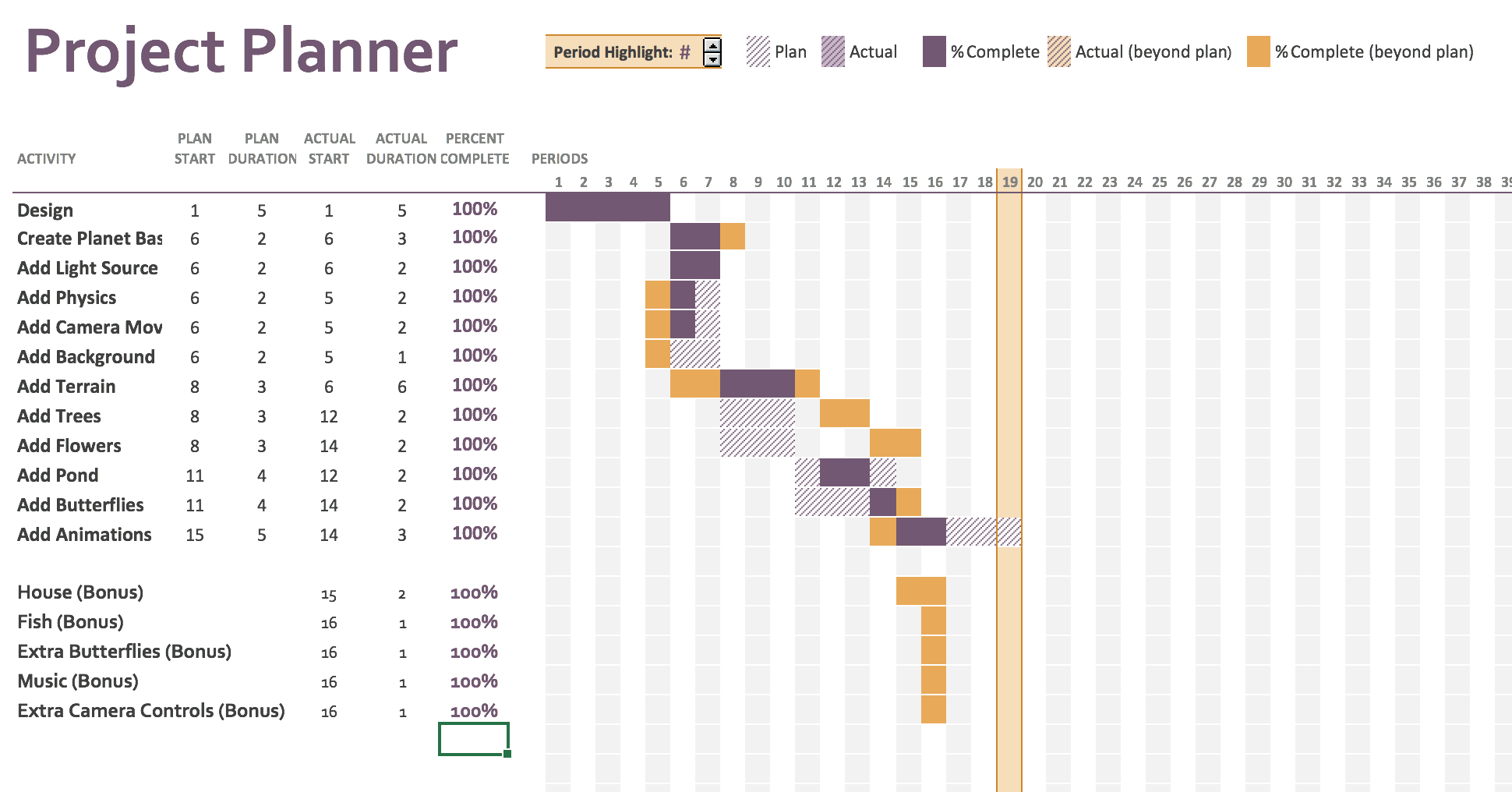
1. A “little” sphere planet that has a radius of 10 (based on Unity’s units) and has a center of gravity coming from the core of the planet. The planet has surface detection so that objects do not go through the surface due to the pull of gravity. Planet rotates around its Y axis.
2. A grass-like terrain the covers all of the planet that is not covered by other objects. The grass has a cartoon-like texture and uses a bump map to make the grass stand out.
3. 8 patches of flowers exist on the planet (10 flowers per patch minimum). Each patch has a color associated with it. The flowers transparency, and texture are cartoon-like. The flowers cast shadows. Each patch has a unique set and arrangement of flowers.
4. 6 trees all with a unique shape, size, and color (can be hard to tell but yes they vary in color). They have a cartoon-like trunk, all with different patterns. They all cast shadows and have collision detection.
5. There is a pond that is at least 3 Unity units deep and at least 5 Unity units in length (from at least one end, to another). The water is cartoon-like using a cartoon texture, and a blue highly transparent color. The water slowly moves in a set direction and reflects light.
6. 13 butterflies, each with different coloring. Each butterfly flies close to the surface of the planet. Each butterfly has a flying animation that closely resembles a real life butterfly. Each butterfly casts a shadow (cannot see shadow for the large dark butterflies because they are created using a skin mesh and Unity Pro is needed to see the shadows). Each butterfly has a unique size.
7. A sun with a light flare that appears to be orbiting the planet.
8. A dark-starry background that surrounds the scene.
9. The user is able to to move the camera around the planet by using the W, A, S, & D keys. W for moving forward, S for moving backward, A for moving left, and D for moving right. The user may also use the Z and X keys to zoom in and out. Z for zooming in, and X for zooming out.
10. Project runs at 60fps on average.
11. Only takes about 3 seconds to start the project.

List 4: Implemented Bonus Features

1. A house with a realistic red brick texture used on the roof (with normal map), a chimney that has black smoking coming out of it, a lantern that emits light, windows that have color, transparency, and reflection aspects.
2. 3 fish, all with unique size, swimming in the pond. Each casts shadows, has a swimming animation, and reflects light.
3. Background music – Animal Crossing New Leaf 7PM Track (on loop).

**Schedule (Completed)**

I have completed every requirement and have implemented bonus features. My schedule is shown below in Picture 2: Little Planet Schedule.

Picture 2: Little Planet Schedule

**Final Project Requirements**

Table 1: Project Requirements and Features shows how my final project meets the class’s requirements.

Table 1: Project Requirements and Features

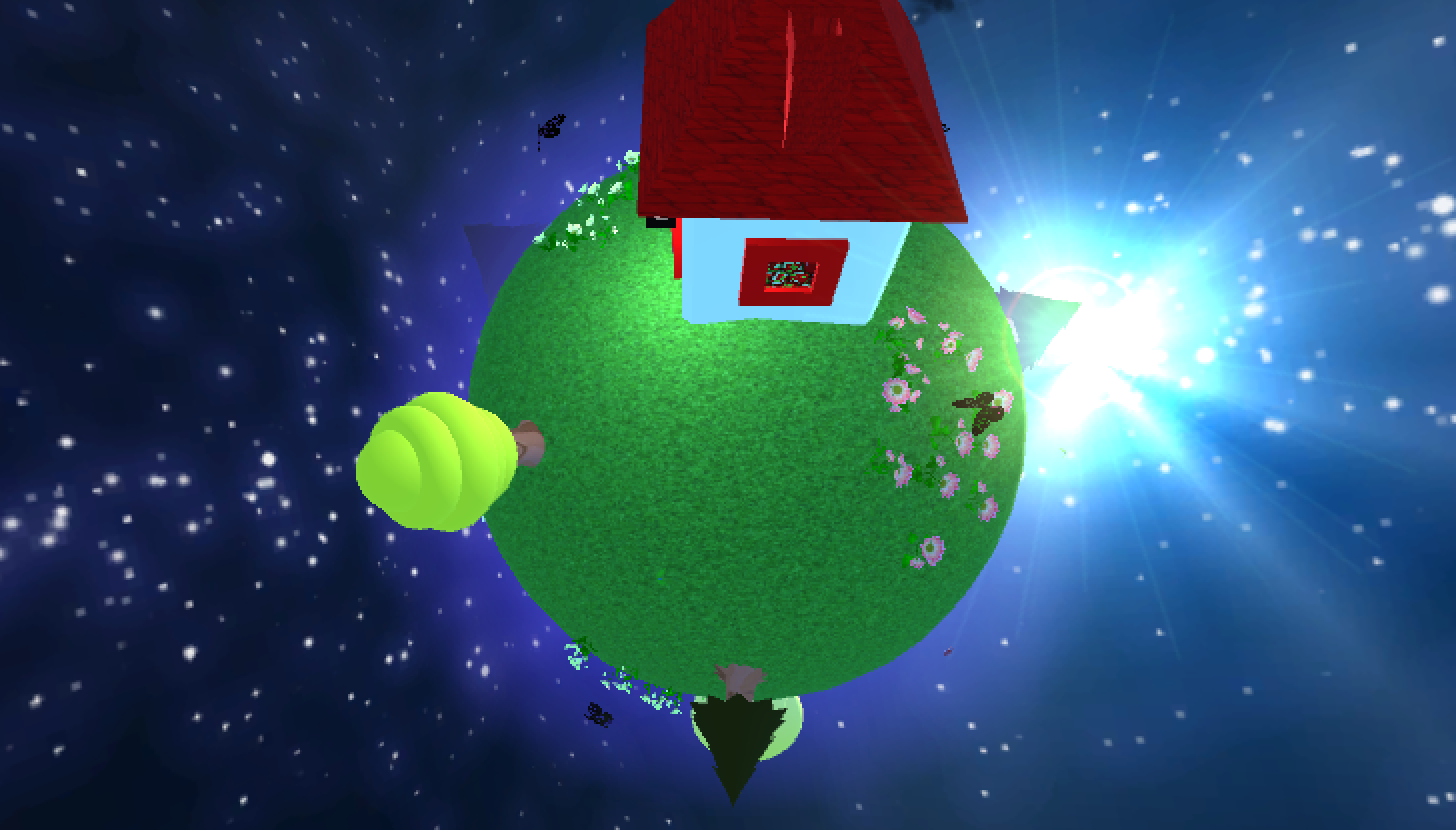
|  |  |
| --- | --- |
| **Requirement(s)** | **Feature(s)** |
| Use colors and transparency | All 3D objects created have their own color, and transparency |
| Use transforms and rotations | Rotation of the planet, and butterflies. Butterflies also translate up and down. |
| Use cameras and lights | The sun and the lantern on the house. Plus there is quite a bit of ambient light to light up the dark side of the planet. The camera is controlled by the user. |
| Use textures/reflections | Many objects use textures including normal maps, and many objects reflect light. |
| Use animation | Butterflies and fish. |
| Use surface recognition / solid modeling | Used but it is not demonstrated in the project. It will be demonstrated more when I add a character that can jump, but that will be after this course. |
| Use or model physical world features | Gravity will be centered at the core of the little planet. Water like physics used for the pond. Butterfly flying animations. |
| Create simple objects directly using primitives - points, vectors and meshes | Almost all objects created by myself in Maya. |
| Create objects using a graphics library built in geometrical functions | Unity’s geometrical functions are used to create, add, and manipulate components of each object. |

**Output**

The following photos of the project’s output demonstrate many of the requirements of the final project. The animations, water movement, and planet rotation components must be observed by running the project since a photograph cannot capture their features. Although the planet gravity feature works and has been implemented, there is nothing done in the project to demonstrate the feature. In code drop 1 I had a cube fall to the planet to test that the feature works, but this feature is not demonstrated in the final version since no objects fall to the planet. My goal is to add on to the project (after the course of this class) by adding a user controlled character that will have jumping abilities, which will involve the scripts created for gravity.

Picture 3: Output Photo 1, Demonstrates the use of the camera and lighting (sun flare). Also, notice the use of the brick like texture on the roof and chimney of the house.

Picture 3: Output Photo 1



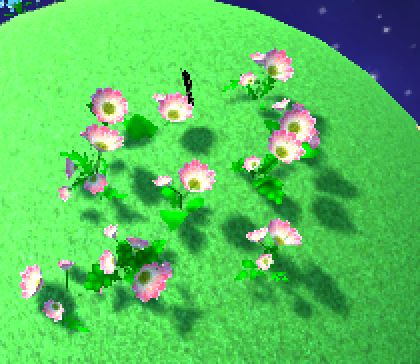
Picture 4: Output Photo 2, Demonstrates the transparency and reflection of the glass material used on the windows and lantern. Also, notice the light coming from the lantern.

Picture 4: Output Photo 2



Picture 5: Output Photo 3, Demonstrates the shadows of the flower objects. Note that the darker butterflies (as shown) do not cast a shadow since Unity Pro is needed to cast shadows for skin meshes. Also note that even though I did not model these flowers, it was very time consuming placing each flower on a spherical surface, and that each patch of flowers is unique and non duplicated (8 patches total, with over 80 flowers).

Picture 5: Output Photo 3



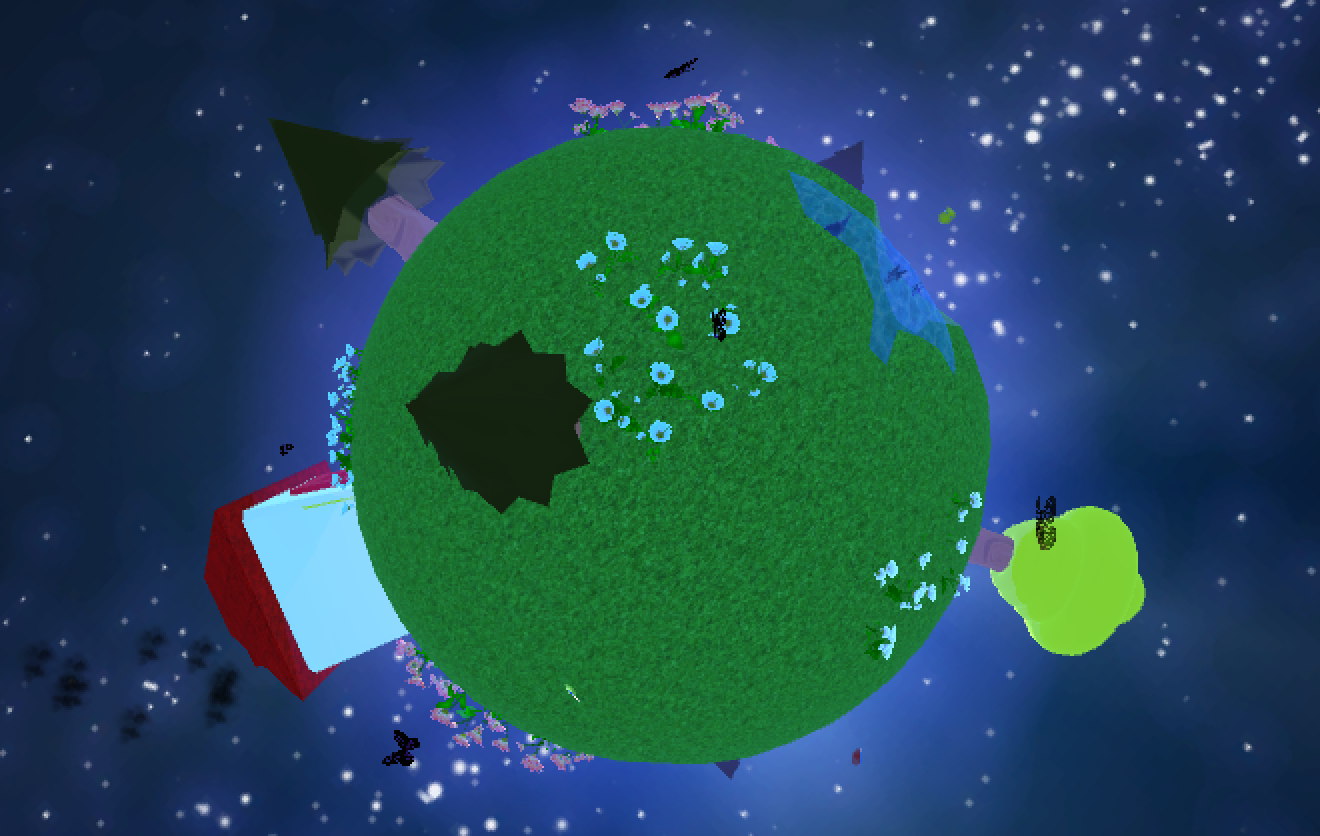
Picture 6: Output Photo 4, Demonstrates the shadow created by the common butterflies that I modeled and animated myself (the shadow shows up since they are not made from a skin mesh like the darker butterflies). Also, notice the shadows made by the tree, and that each tree has a different bark pattern for their trunk.

Picture 6: Output Photo 4



Picture 7: Output Photo 5, Demonstrates the blue glow around the planet, and how even the dark side of the planet is visible due to ambient lighting (although it is not realistic, it makes for a better aesthetic look). You can also see how the pond is transparent, has a cartoon water texture, and curves with the surface of the planet.

Picture 7: Output Photo 5

****

Picture 8: Output Photo 6, Demonstrates the transparency of the butterflies, the cartoon grass terrain, and the zoom in capabilities of the camera (by pressing Z).

Picture 8: Output Photo 6



Picture 9: Output Photo 7, Demonstrates the zoom out capabilities of the camera by pressing X (note how the camera is always focused on the center of the planet).

Picture 9: Output Photo 7



Picture 10: Output Photo 8, Demonstrates the varying size, shape, and materials of the trees. It also shows the fish in the pond and how the pond reflects light.

Picture 10: Output Photo 8

