1 Introduction

For the project I have made my own file format, cgworld. It is a binary file format which can store both static and dynamic objects, and supports textures and lightmaps. By reading the class overview you will automatically see how the file format works.

The file itself was created with a little convert tool I wrote, the world itself was first created in 3D studio MAX.

Using the keyboard arrows you can move around. By left clicking the camera is activated and you can look around with the mouse. Space stops the animation (and walking as well). Pressing enter shows some statistics.

2 Class overview

2.1 3D math classes

BoundingBox

A bounding box is a box in 3D space with its faces aligned to the three axes. It is used to determine the space some object takes roughly. A bounding box of an object will always completely contain the object, but obviously empty space is also included. The bounding box class contains a minimum and maximum point in 3D, which together define the box.

Plane

Defines a plane in 3D space by storing an origin and a normal vector.

PointAverager

Can average a set of point to get the average point of the set.

Ray

The Ray class represents a ray in 3D space, that is an origin with a direction vector. The direction vector can have any length.

Triangle

Triangle represents a triangle (three vertices).

Vector3

Vector3 represents a three dimensional vector.

Vertex3

Vertex3 represents a three dimensional vertex (point).

2.2 Helper classes

AutoArr<T>

The AutoArr template class can store an array of any type T. It does not automatically grow but it does automatically free the memory allocated for the array when the object is destroyed. It allows the user to get a direct pointer to the memory. Reallocation is possible but it does not copy existing data into the new memory.

DataFile

DataFile is a simple serialization class. It can either read from or write to a binary file. It has several overloads of the *exchange* method to load or save simple values and arrays. More complex classes should write an *exchange* method to (un)serialize its data. An exception class *DataFile::Error* is used to indicate errors that occur.

DisallowCopy

This class prevents a class from automatically creating an assignment operator. This is dangerous with pointer members so by inheriting from this class this copying prevented.

PtrList<T>

This template can contain a dynamic array of object pointers. All object pointers are automatically deleted when this object is destroyed. It can also be serialized if the objects stored implement an *exchange* method.

2.3 Space partitioning classes

CGFaceID

A face ID is a unique reference to a face in the cgworld. It consists of an object index and a face index within that object.

CGFaceTree

The face tree is used to partition the static world into a two dimension tree. It is an Axis Aligned Bounding Box tree (AABB tree). Each node consists a bounding box in which all faces it covers are in. Each non-leaf node does not have any faces but has at most two child nodes. Its bounding box is the bounding box of its childs' bounding boxes. Each leaf node has a set of faces. When the world is drawn, the tree is traversed. When a node's bounding box is not visible it can be discarded, including all its childs. This is a very efficient method to restrict the number of faces drawn. The cgworld's static faces are all stored in this face tree.

CGFaceTreeNode

CGFaceTreeNode represents one node in the face tree. As said it has a bounding box, and to represent the face list it has an index in the tree's face list as well as a count that tells how many faces it possesses. Non leaf nodes have zero faces.

NodeSelector (abstract class)

This abstract class has one important method *includes*, which takes a bounding box of a node in 3D space, and should return whether this node should be selected or not. For example, the AABB tree uses this to determine whether a node is visible or not.

BoundingBoxSelector

The bounding box selector implements the NodeSelector class. In its constructor it takes a bounding box. It will select a node if its bounding box intersects the bounding box given to the constructor.

TriangleIterator (abstract class)

A simple iterator that iteraties over triangles. It is abstract and should be implemented by other classes.

CGFaceTreeIterator

This class implements TriangleIterator, so it can iterate over a list of triangles. It takes a NodeSelector in its constructor and will iterate over the face tree using this node selector. The iterator uses a simple state machine to iterate over the faces.

CGRoom

A room is a part of the world just like a physical room. It is used for additional space partitioning by the portal engine. It is defined by a set of planes that represent the walls. Currently each room has 4 walls in this project. Rooms may overlap a bit (to prevent some problems).

CGPortal

A portal connects two rooms. For this project the portals have 4 points that define them. From one room, you can look through a portal to another room.

PortalEngine

The portal engine determines which rooms and which parts of it are visible. It does this by looking at the room(s) the user is currently standing in, and then looking at all portals of that room to see whether we can see other rooms through it. This is a recursive process since we may see a portal through another portal. The PortalEngine class can give a PortalNodeSelector that will select only nodes visible through portals.

PlaneClipper

A plane clipper contains a set of planes. It can tell whether a bounding box is (partially) inside the planes.

PortalNodeSelector

Selects node based on a frustum and a set of PlaneClippers. If a node is in the frustum or within any of the PlaneClippers, it will select the node.

2.4 World classes

Camera

The camera class represents a camera in 3D. It has a position vertex and yaw and pitch looking angles. Methods are supplied to move the camera and look around. Collisions are also tested by the camera using the CollisionTester class. It also takes care of gravity.

CGObject

A CGObject is an object in the cgworld file. It has a set of vertices for its triangles, normals, texture coordinates and lightmap texture coordinates. It can also refer to a normal texture and a lightmap texture (CGTexture objects). Finally it stores its color (ambient, diffuse, specular) and can be transparent.

CGDynamicObject

This class is a dynamic version of CGObject. In addition to the properties of CGObject, it has a name, position, yaw/pitch/roll and texture shift. Its position and rotation can be changed to anything so that it will appear in a different way. Texture shift consists of two relative s and t values that are added to the texture coordinates. Dynamic objects have some limitations. One is that no collisions are tested against its faces. Another thing is that they are not added to the AABB tree, only basic visibility testing is done.

CGPath

CGPath represents a path in 3D by storing a set of points. Paths are always closed. A position can be extracted from the path by giving a value between 0 and 1, indicating the position on the path wanted. Points in between the stored points are interpolated.

CGTexture

To define the textures used in the cgworld file, CGTexture objects are used. They simply contain a filename of the texture to be used and some methods to activate it. Mipmaps are automatically generated for each texture.

CGWorld

CGWorld is the class that binds all the CG* classes together. One CGWorld object is created for each cgworld file and all data can be accessed through this object. It stores:

- All CGObjects
- All CGDynamicObjects, accessible by name
- All rooms, accessible by ID
- All portals, accessible by ID
- All paths, accessible by name
- All textures, accessible by filename and ID
- The face tree

In addition to these, which are all stored in the cgworld file as well, it also has a Camera, Timer, PortalEngine, FaceRenderer and SkyMap object. The render method will render a single frame of the whole world.

Dino

The whole 3D world is created using the cgworld file, but the dinosaur is an exception because it is loaded from an SGF file. This Dino class takes care of the loading and movement of the dinosaur.

CollisionTester

This class can be used to test for collisions. The user is represented as an ellipsoid. All coordinates are first translated to ellipsoid space so that collision can be tested using a simple sphere.

FaceRenderer

All faces in the cgworld file are rendered using the FaceRenderer class. This class takes all the face data an object gives it and renders it, possibly with textures and lightmaps. Also, it can queue transparent faces because they needed to be drawn last.

Frustum

The Frustum class can extract the viewing frustum from the openGL matrices so it can be compared with objects to see whether they are visible.

Model

The class to load SGF models, as used in exercise 2. Now it is also possible to add a texture to a model.

ModelPart

The class for one part of the SGF models, as used in exercise 2 It now also has texture coordinates and can do a simple box texture mapping.

SkyMap

SkyMap draws a skybox to fake an environment. It simply draws 6 faces of a box with the right textures on them.

Light

Very basic wrapper around an openGL light.

2.5 Application classes

InputManager

The input manager dispatches all keyboard and mouse events to the right places.

Timer

This class represents a timer that can be started and stopped at any time, and can tell the user how much time has passed.

Statistics

Singleton to gather statistics of the application and build a string showing them all.