Project 3 Final: A (Tiny) Amusement Park

Introduction

Interactive graphics is the process of creating a program to allow users to interact with the virtual world created by you. In this project, we would like you to create an amusement park allowing users to have fun and interact with the program. You can use your creativity to create an interesting environment or an interactive simple game.

The Basic Task

This project will provide you with experience in modeling objects and creating interactive animations for computer graphics, and introduce you to many more of the features of OpenGL. Your goal is to make the scene a theme park based on the roller-coaster.

The overall goal of this project is to give you an opportunity to explore topics in interactive graphics: how do you make things that look interesting, and are interactive. While some of this is artistic (you need to pick interesting objects to make and good textures/... to look nice), some of it is technical: you need to pick things that can be implemented efficiently and have interesting behavior. Like project 1 and 3, this project defines a set of sub-goals with points awarded for each goal. Unlike project 1, the goals are far more loosely defined, so there is scope to try interesting things to get all the points available.

In terms of your grade, effort spent on technical are more valuable because we are computer scientists. For example, it is better to spend your time making a simple "blocky" car drive around in an interesting way, or to make a simple shaped car out of parametric surfaces, or to light the car in an interesting way, then to carefully model a gorgeous model of a car. (of course, if you want to make model a gorgeous car, implement Bezier patches to display its curved body, have it realistically race around a track ... - we won't complain).

Please print Project3-Grading.doc, let TA to check your score.

Some specific things we want you to learn from this assignment (which will explain some of the requirements):

1. To try out some of the technical topics that we've discussed in class (subdivision surfaces, culling, ...) or topics we won't discuss too much in class (particle systems, fractals, ...)
2. To get some experience with how textures are used to make simple objects look more interesting.
3. To get some experience with creating geometry for graphics.
4. To gain experience working with a larger, more complex graphics application.
5. To gain some experience creating the behavior/motion of graphics objects.
6. To work with shaders.

The Tasks

Each task requires modeling one or more objects using a specific technique from class. The points available for each technique varies according to the difficulty of the task. In all cases, you get a base number of points for implementing one object with a technique, then an extra 1 or 2 points for each additional, but distinct, object with the same technique. You can score points for a maximum of three objects with any one technique. For instance, If an object involves more than one thing, such as a texture mapped, swept surface, then you can score points for both texture mapping and sweep objects.

Total = (Check Point #1 + Check Point #2 + Final Demo)
The maximum number of points is 110. As with Project 1 and 2, you can do as much as you like, and we will truncate to 110 as the final step in computing the grade. The individual tasks, point value, and example objects are:

Technique	Requirement	Points	Suggestion
Texture Mapping	Add texture mapped polygonal objects to the environment. Each "object" for grading purposes consists of at least 5 polygons all texture mapped. Different objects require different maps.	3	Buildings, walls, roadways Hierarchical Animated Model
Performance Tricks	**Level of Detail** (3): Draw objects with different levels of detail - if they are far away, draw a simpler version. Showing this off is hard because if it works correctly, no one will notice (except that your program might be faster). Have a	15	Foundation, firework, ...

	UI to force different levels of detail so we can see them, and see the performance difference.		
Shadow Mapping	Shadows on the ground are easy. Shadows cast from one object onto another are much harder. Implementing shadow mapping (or shadow volumes) is one way to do this. 1. Directional light and Spot light (2.5) **2.** Point light (2.5): The depth map we need requires rendering a scene from all surrounding directions of a point light and as such a normal 2D depth map won't work; you need to use a cube map instead.	5	
Really Cool Shaders	Graphics Processing Unit(GPU) has become very important aspect in graphics. We would like to explore the usage of them such as environment map, particle simulation, water simulation, and so on. Everyone has to write 3 shaders. But if the shaders do something really cool, that counts for technical challenge. Bump Mapping definitely counts as a technical challenge. A properly anti-aliased procedural shader would be a technical challenge. Phong shading would be a technical challenge except that you can get the code from just about anywhere - combine it with something more imaginative to make a technical challenge. We'll give technical challenge points for really imaginative shaders.	15	Foundation,firework, ...
Hack Rendering Tricks	**Local lights** (1): have a light that only effects nearby objects. You can't just do this using the OpenGL falloff since that limits the number of lights you have - you'll need to switch lights on and off depending on what object is being drawn. Note: local lighting is NOT the hack "spotlight cones" that my sample program does. Consider putting a flashing (or even spinning) siren on a police car,or ... **Inter-object reflections** (3): Reflections of actual (dynamic) objects are really tricky - as opposed to using environment mapping with static environments. Implementing shadow mapping (or shadow volumes) is one way to do this.	15	...
Non-Photorealistic Rendering	Give your world an artistically styled look to the drawing. For example, make everything look like a pencil drawing by tracing object edges and making things squiggly, or use "toon shading" to make things look like a cartoon. Note: if you are really going to do an NPR world, we might be willing to remove the texture requirements - but only if you'll be doing enough NPR stuff.	15	Foundation,firework, ...
Very Advanced Texturing	**Skybox** (3): make a textured sky - have clouds and stars (at night). But note that a proper skybox stays fixed relative to the camera (so it doesn't have to be so huge that it causes Z-Buffer issues). **Billboard Object** (3): model a complex shape by using a flat object that moves to face the viewer (and probably transparency). Nice trees can be done this way. **Projector Textures** (5): make a slide projector or something that creates an unusual effect. To get full credit, it needs to be clear that you are using the texture matrix stack to get a projection. **Environment Map** (3): Use environment mapping to create a reflective surface. Be sure to describe how you made the map.	15	...
Hierarchical Animated Model	Add a hierarchical, animated model. The model must combine multiple components in a transformation hierarchy. Different models need different hierarchies.	1~5	Ferris Wheel, any number of other fairground rides.
Paramatric Instancing	Add an object described by parameters. You must create multiple instances with different parameters, and each class of model counts for separate points, not each instance.	1~5	Trees (cones on sticks), buildings, even rides
Sweep Objects	Add an object created as a sweep, either an extrusion or a surface of revolution. The important thing is that it be created by moving some basic shape along a path. The overall object must use at least three different uses of the swept polygon. In other words, something like a cylinder isn't enough, but something like two cylinders joined to form an elbow is. Moreover, this technique requires a user interface to control the extrusion or surface of revolution, such as selecting the face to control, adjusting the length of the extrusion, or progressively displaying the rotation process of the surface of revolution.	1~5	Rails for the roller-coaster, trash bins, trees
Subdivision	An object defined using the modified butterfly scheme. You must include a key press that refines the model, so that we can see the improved quality. The sphere example from classcan help, somewhat, with this.	1~5	The roller- coaster car, organic looking roofs, ...
Other Modeling Methods	**Complex Procedural Model** (3): Generate complex shapes using a procedure. You might make branching trees, buildings with lots of columns, fractal mountains, ...	15	Foundation,firework, ...

	Fractals (3): Implement fractal geometry (a random kind of subdivision) to make some complex shapes like mountains or trees. **L-Systems or other plant generators** (3): These are procedural models like fractals that have specific rules forgenerating trees.		
Animation Techniques	**Particle Systems** (1~2): Model a complex, moving object as a set of little particles. You can make fireworks, rain, snow,fountains, fire, ... **Fake Physics Effects** (1): Use some kind of math techniques to make "physical" animations (like flags waving or water ripples in a pool, or flames). Rather than trying to simulate the physics, experiment with using procedural "hacks" (a waving flag can be a sine wave ...). Bonus challenge for doing fake physics using the shaders (for example, having the flag movement or water waves generated in the vertex shaders). **Very Complex Behaviors** (1): Some behaviors are so complicated that they constitute a technical challenge. For example, flock simulation is a technical challenge. **Complicated Animations** (1): a human walk around.	10	Foundation,firework, ...

If there's something that you want to do that you think is a good task, but not listed, please ask. We may extend this list at a later date as we think of more ideas. We may not be able to help you with some of these, so doing it will require some bravery and determination.

It is important that if you do something, you are able to show it off in the demo/make a picture for your album. So if you model some nice object, make sure there's a fast way to get the camera to go there. Or, if you do subdivision, show different levels of the same object so we can tell you really did the subdivision.

Some of the challenges require something to be complicated. You might wonder "when is an animation complicated enough that it qualifies as a challenge" or something like that. Generally, if you have any doubt, then it probably isn't so complicated. But if you are in doubt, ask.

Components

Your park must have:

1. Multiple objects moving at any time
2. Multiple different types of behaviors
3. Multiple different types of buildings / scenery
4. Multiple new textures. Some must be hand painted. Some must not be flat (that is, it must wrap onto multiple polygons)
5. You must attempt "enough" technical challenges (see the technical challenges page).
6. You must have at least 3 shaders in your program (by "shader" we mean a pair of vertex/fragment programs attached to an object). At least one of these shaders must provide a procedural texture, and at least one of the shaders must be (properly) affected by the lighting. At least one of the shaders must be affected by the time of day (so you need to figure out how to pass the time of day to the shader).
7. Your program must work at a sufficient frame rate (which isn't hard since the computers are so fast).
8. You must add something that is affected by the time of day (besides the one shader used to fulfill the requirement above). For example, you can have an object that changes color (the shader is sensitive to the time of day) and shape (something besides the shader is sensitive to the time of day).
9. You must use at least one type of "advanced" texture mapping: multi-texturing, projective (slide projector) texturing, environment mapping, bump mapping, or shadow mapping. (If you want to pick something not on this list, you may want to check with us to make sure it counts)
10. An object made out of a curved surface. You can implement subdivision, or some form of parametric surfaces, or do a surface of revolution, or ... This is described more on the technical challenges page.

The Program Skeleton

Your program should build based on the roller coaster project

Hints and Suggestions

1. Have fun and be inventive.
2. A key thing to consider is polygon count. Graphics hardware can only display so many polygons in a second, and if you try to display too many the frame rate will collapse. Texture maps also use memory, so too many textures can even more dramatically affect performance.
3. The way the train alignment is set up, it is simplest to do just a single carriage, and a short one at that. Doing lots of cars makes it harder to keep then on the track, although it is possible.
4. Make use of the OpenGL error checking mechanism. It is described in the OpenGL Programming Guide.
5. Start simple - just try to get a polygon to appear in the center of the world.
6. The way the current carriage transformations are set up, the origin for the train is assumed to be atthe bottom (at track level).
7. It is OK to have multiple modeling techniques in one object. For instance, you could have a carriage made up of some texture mapped polygons with some subdivision areas. You get all the points if you do a sufficient amount of each technique.
8. It is OK to borrow code from other sources - but not other students. You will probably learn as much trying to figure out how someone else's code works as you would doing it yourself.
9. Texture images abound on the web, so feel free to use them. Or you can use a program like Photoshop to create your own. You might even find a use for the first project.

What to hand in

1. As usual, you must hand in everything needed to build and run your program, including all texturefiles and other resources.
2. If you work with a partner, only turn in one copy of the project. In the other person's directory put asingle file in your handing directory - a README.txt that says where to look.
3. In your readme, please make sure to have the following (you can break it into separate files if you prefer):
    1. Instructions on how to use your program (in case we want to use it when you're not around)
    2. Descriptions of what your program does to meet all of the minimum requirements.
    3. A list of the objects you modeled (if you made lots of different objects, just list the 5-10 mostinteresting ones). Please order the list so the most complicated/impressive one is first.
    4. A list of the behaviors you made. Please order the list so the most complicated/impressive one is first.
    5. A list of the shaders that you made with a brief description of each.
    6. A list of the technical challenges that you attempted / completed, with a description of what youdid and what you used it for.
    7. If you used the sample, code, a file that describes any changes you made to the "core" of thesystem (e.g. other than changing main.cpp and adding new Objects and Behaviors).
    8. If you did not use the example code, an explanation of why you chose not to, and a discussionof your program's features.
4. You should make a subdirectory of the project directory called "Gallery." In this directory, please puta few JPG pictures of the best scenes in your town. Please name the pictures login-X.jpg (where Xis a number). Put a text file in the directory with captions for the pictures. (note: to make pictures, use the screen print and then use some program to convert them to JPG).
5. You must also make a subdirectory of the project directory called "Video" to put a 1~2 minutes ofvideo capturing your world.