Dec 04, 2012

UPDATED PROPOSAL

Christian Zommerfelds
User Name: cgzommer - ID: 20493973

Note: the objectives of this project were partially discussed with Prof. Stephen Mann

CS 488 - Final Project - Shape Twisting Puzzle Game

Purpose :

The design and implementation of this project will give me a more in depth understanding of the subjects covered in class and of other computer graphics topics by using them in a creative way. This project consists of implementing a 3D puzzle game with different graphics features. Topics of this project will include some of the topics from previous assignments, and also the following:

- Texture mapping
- OpenGL shaders
- Mesh generation
- Deformable shapes
- Particle systems
- Animation

Statement :

Project Description

The projects consists of developing a simple 3D single player puzzle game where the player controls a robot and has to help other less intelligent robots traveling from one end to the other in the level. The game's main concept is that there is a long connected floating object, which will have characteristics like the Möbius strip, on which the player can walk on and has to accomplish his tasks. The player can stand and move on all sides of the object, but to get from one "side" to the other, the player is not allowed to cross an edge. The floating object the player walks on will be a solid volume and its cross section will be a square. There will be specific parts in the object (depending on the level) where the player can twist the floating object around its axis so he and the other robots can reach other sides of the object. On one end of the floating object there will be a starting point for the little robots and one ending point. The main character can activate the little robots and they will simply always go straight until they hit an obstacle or the finish point. Hence the player's task will be to prepare the twisting of the floating object so the little robots can pass in one go.

The player will be able to walk forward and backwards on the floating object. On "branches" in the path, the player can chose a direction to go. He can control when the main robot twists parts of the floating object and when the little robots are activated. A third person camera which rotates automatically to the players orientation should be implemented. To understand the level, the player could rotate the camera around the main robot by moving the mouse. Zooming in and out should be available using the mouse wheel.

The game will feature different computer graphics effects. The floating object will be texture mapped. It should be made to look somewhat surrealistic. A glow shader and a particle system will help emphasize this style. When the main robot and the little robots perform actions, their bodies should be animated. As a post processing step, a motion blur shader should be implemented. The effect should be visible when the player rotates the camera or moves along the level.

In addition to the graphics, sounds for the character's actions and an ambient sound will be added to the game.

Implementation

I will start with the implementation of the hardest objective: the generation and rendering of the floating object. First I'll try to get it working without twists, then add the twists to the data structure and rendering.

Then, I'll create a main character model and make it move along the paths using correct orientation. Next will be the implementation of the third person view and interface.

After that, add visual effects to the floating object: texture mapping, particle system, motion blur and glow.

Add animation and sound. As last, work on the polish of the game.

Challenges and Motivation

The shaders could turn out to be a challenge because I never developed one of them before and I don't have the experience about how complicated it is to write one and to integrate it in an OpenGL scene without any shaders. Possible other parts like texture mapping would have to be rewritten because of the new shaders.

A second challenge will be the generation of the meshes for the twisted object out of the extended 3D Bézier curve data structure, especially for the joints (i.e., when three paths come together), and keeping track of the twist transformations of the vertices. It will require special attention to get it right.

I was always very interested in creative game development and in developing new game concepts. This is the reason why I will take this opportunity to create a game. From the graphical point of view, I have special interest in how to program "special effects" like glow and motion blur with shaders.

Technical Outline :

Modeling

The robots have to be modeled and stored into a data file. This will be accomplished using Lua, as it was used for previous assignments. New functionality has to be added to fit the modeling needs for the robots (e.g. quads and textures). The joints will be labeled so they can be animated later on. The player's robot should have a different model than the small robots. Each of the small robots should have a unique visual feature, i.e., different shading, which will be generated procedurally out of the models. A start and a finish object have to be modeled.

Animation

To implement the robot's animations, a data structure containing the animation state and the information about the angle of each animated joint has to be created. When the game gets updated, the animation has to be updated too and joints moved accordingly. The animation for the main and little robots will be hard coded in the program (i.e., not loaded from external data). There should be at least 3 different animated actions: the main robot moving, the little robots moving and the main robot twisting the ground object.

Mesh Generation and Transformation

The mesh of the floating object where the player's character walks on will be generated from a special data structure. This data structure will consist of 3D Bézier curves with added data for keeping track of the orientation (or up vector) of the floating object around it's own axis. Additionally the ends of two curves with same orientation can be linked together so the game knows they belong together and that the player can move from one curve to the next. If 3 or 4 curves are connected together with the right orientation and angles, they will form a branch where the player can decide which way to go. When branches are detected, smooth transitions (no 90° edges) should be generated.

To generate the mesh, the following algorithm will be used: start at on end of a curve, get the up vector of that end, and generate four vertices forming a quad. Continue along the curve by a small offset (defined by how fine the subdivision should be) and calculate the new up vector and add another four vertices according to the orientation. Connect those vertices with the vertices from last step. Continue until it reaches the end of the curve. To make the joints, detect if there are more than two curves connected, and get the one that got in at 90° with the other two (no other angle is allowed). Add a rounding to make the joint smooth.

Locations where the player can twist the floating object are in connections between two curves. When the player twists the object, the orientation vector is modified at the connection point and the mesh is updated again. There should be a limit in the amount of twisting (for the logic of game, it only needs 4 positions, then the player can get from any side to any other).

Texture Mapping

The floating object where the player's character walks has to be texture mapped. Each side should have a different texture or colour overlay. When a join gets twisted, texture blending should be used to make a transition to the new colour. This should be accomplished using alpha mapping (i.e., overlapping textures with an alpha layer to know which texture is more visible).

Particle System

The particle system will be used to add a better look to the floating platform. Particles will be released from the floating object to give it a "smoky" look. The particles will be drawn using a soft texture. No particular gravity direction will be used for the particle simulation, as the player should have the feeling that the floating object it is the gravitational center. The particles will simply move away from the floating object in a randomized fashion and fade out.

Third person view

The camera has to follow the player and keep him at the center. Each mouse axis will move the camera in one direction. The camera's horizon should always be aligned with the spot where the player's character is standing. This means that when the robot moves around the object, the camera should update its orientation.

Shaders

Two shaders will be added to the game to add more interest to the scene. Both of the shaders are pixel shaders. The glow shader will add a glowing effect around the floating object. The way it works is it takes the fragments of the objects and performs a blur effect and make it a light glossy colour. The original object is then rendered on top. The blurred, coloured object will be seen as a glow.

The motion blur shader will make the moving around the level more dynamic. It will be implemented as a pure post processing shader. The way it will work is it will get Z-buffer values from the vertex shader and the view-projection transformation from the current and last frames. With those data we can calculate the current and old position in the scene of the pixel and find out the velocity by calculating the difference. Then use the velocity to do some pixel averaging in that direction, which will give the rendering a motion blur in the direction of the movement. As Gilberto Rosado describes in his article [3], this is a fast and modular design for motion blur, instead of using copies of the rendering frame.

Sound

Calm background sound/music should be played during the game. When the player moves, twists, or activates the little robots, different sounds should be played.

Bibliography :

[1] Hubert Nguyen, GPU Gems 3, Upper Saddle River, N.J.: Addison-Wesley, 2008
Chapter 27. Motion Blur as a Post-Processing Effect, Gilberto Rosado, Rainbow Studios

[2] Greg James and John O'Rorke, Real-Time Glow, Gamasutra, 2004
< http://www.gamasutra.com/view/feature/2107/realtime_glow.php >

[3] John van der Burg, Building an Advanced Particle System, Gamasutra, 2000
< http://www.gamasutra.com/view/feature/131565/building_an_advanced_particle_.php >

Objectives

Full UserID: ________________________ **Student ID:** ________________________

___ **1:** The main robot, small robots and other game objects have been modeled and display correctly. Each little robot has unique shading.

___ **2:** Animation of the main and small robots are displayed correctly and are triggered by the player's according actions. At least three different animated actions have been created.

___ **3:** Implementation of the floating object data structure using 3D Bézier curves. Rendering of the generated mesh works. Joints are generated and look smooth. Game can load and generate the floating objects (i.e., the level) from different level files.

___ **4:** Player can twist the floating object around its own axis in real-time at specific positions in the level. The object looks solid and does not shrink.

___ **5:** Texture mapping on the main floating object is rendered with different appearance on each side. Smooth texture transitions are implemented when two sides with different appearance meet.

___ **6:** Particle system works and produces the desired surrealistic effect on the floating object.

___ **7:** Third person view that follows the orientation of the player works. Mouse can be used to rotate view around player. A texture background is rendered using a sky box or sky dome.

___ **8:** A glow effect is visible around the floating object. It can be turned on and off and glowing properties like size should are modifiable in the menu.

___ **9:** The motion blur shader produces blur when objects move rapidly. User can deactivate or increase the effect of the shader.

___ **10:** Sounds play at the right time and ambiance music is present.

Appendix : Visualization of Game Logic and Concept Art