CS488
A5 Proposal Revisal

"Please, he's no different from the rest of you organisms.
Shooting DNA at each other to make babies… I find it offensive!!"
- Bender [Futurama]

Topics

- Displacement Mapping
- Transparency/Alpha
- Texture Mapping, Bump Mapping with animation
- Keyframe Animation

Statement

You awaken after unknown time; you hear a voice call to you, a beacon. You are a single sperm, but quite unlike your sibling cells. You are mutated; with abilities not possessed by the others, and you and the ovum have the potential, if joined, to form a new and better being. If the ovum joins with another, despite its own mutation, the chance will be lost. You must advance through the reproductive landscape, avoiding the dangers that beset you. Normally the usual defenses of the body do not attack the foreign sperm cells, but you are so different, that you have been designated a threat. No less so than the other sperm, who do not recognize you as being from the same originating genes.
You must defend yourself, advance, find new mutations, food, and flagellate your way to the Ova, who must accept the first to reach her, and you must be that one, to form the

Zygote

This will be an interactive game based on OpenGL using the SDL stub. You control the sperm and progress through the game as based on advancing levels, with puzzles, and 'boss' creatures (as development permits)

Goals

- Keyframed animation for the various entities will be implemented using an extended A3 program. The walls of the environment will be bumpmapped, and the map will animate to suggest an undulating environment.

- Alpha/Transparency will be implemented to make some of the entities look more fluidic/cell-like, and to enable modeling of some of the environmental elements. The liquid you are presumed to be in throughout the course of the game will be transparent, but haze effects will be implemented.

- Frustum Culling will be used to limit the amount of the environment shown, and the enemies and other entities within in, to the area projected by the camera. This should help increase overall game screen.

- The characters in the game will need to remain within the environment, so static-dynamic collision detection will be implemented. The enemies and their projectiles will intersect with you, and yours them, so dynamic-dynamic collision detection will be implemented as well.

- There will be a Heads-Up Display orthogonally drawn on the screen to convey your current status.

- Particle systems will be implemented to generate shrapnel from defeated enemies and for other graphical effects, such as when enzyme 'bullets' detonate.

Interface

The game is played with the viewpoint being just above and behind your sperm character, and as the character rotates, so does the camera.

- Up, Down, Left and Right keys rotate your character, on the spot, in those directions.

- W and E keys will propel you forward. Either one alone propels you slightly, but with no constant momentum, you must alternate between them to flagellate left and right to build up speed.

- Space Bar will fire the currently selected projectile

- ESC will bring up a configuration menu to view your status in greater detail, switch weapons and upgrades, and toggle special debugging modes. The interface will be with the mouse.

There will be at least one debug mode, to demonstrate Frustum Culling. It will allow the viewpoint to be manipulated in a way similar to A3, using the Left and Middle Mouse Buttons for translation, and the Right for trackball rotation.

Modules

Entity – This will be the base class of derived classes for the various characters in the game, you, enemy sperm, antibodies, etc.

Sound – This provides the interface to the sound library which will be an adaptation of the sound manager provided in the SDL stub.

Viewer – This handles the overall rendering and input handling for the game.

World – This is where the current level is defined, and handles the characters interactions within it.

Algebra – This will be borrowed from previous assignments as the general 3D math library

Technical Outline

Frustum Culling – Environmental polygons that outside the viewing pyramid can be eliminated, this will reduce the overall number of polygons that the system attempts to render.

Collision Detection – Environmental collision detection will use bounding volumes and restricted planes, the faces of the environment need not be too complicated. The dynamic collision detection will use bounding spheres as the characters are mostly spherical anyway (save the flagellum, which is not considered part of collisions)

Displacement Maps – Maps using grayscale will define the contours of the world, where sides are bounded by either vertical walls, or invisible straying barriers. Another displacement map will be used for the 'ceiling', which will allow either liquid-surface tops (flat) or cave-like levels

Keyframe Animation – Characters will be animated using keyframe animation sets from an altered A3 program. The keys will be exported and incorporated with the models, and linearly interpolated.

Animated Bumping – The normals of certain elements of the environment will animate to give an undulating effect. The animation is not intended to look like anything, just to give the impression of being in a living medium.

Alpha – Transparent and translucent elements need to be drawn last after the opaque, and then, back to front. This is due to the z-buffer test losing meaning when drawing non-opaque polygons.

Environment Generation – Environments are formed by a number of planes which have displacement maps applied to them. Simple areas will contain only one displacement-mapped plane for the bottom with a simple plane on top (suggesting the top of liquid media) and other areas may employ two or more planes to form more cavernous areas.

Milestones

- Create environment displacement system with animating normals
- Model and convert characters entities in 3D Studio and export to obj
- Create keyframe animation program and use to animate the characters
- Implement navigation and input handler
- Import the models and begin game design
- Implement frustum culling and haze
- Implement collision detection
- Create rudimentary UI for the enemy characters
- Create graphical and audible assets

Bibliography

OpenGL Technical FAQ
http://www.opengl.org/resources/faq/technical/transparency.htm
General information on implementing transparency, and the reference for the difficulty in drawing order

Efficient View Frustum Culling
Sýkora, Daniel - Jelínek, Josef
http://www.cg.tuwien.ac.at/studentwork/CESCG/CESCG-2002/DSykoraJJelinek/
Useful Document on frustum culling

Organization

A5 – root of project, executable location, and of README

A5/texture – texture maps located here

A5/sound – location of sound files and music files for use with the game

A5/doc – documents stored here, game manual, etc.

A5/src – source files for building project

A5/model – model information (meshes, related information) stored here

Sources

All game source will be stored inside A5/src root.

Executable

The final Executable will be stored in the A5 root directory, and runnable by ./zygote

Data Files

Source images for texturing and deplacement maps will be stored in A5/texture in PNG format.

Sounds will be stored in the A5/sound directory in whatever format required by the 3rd party sound library.

Meshes will be stored in obj files (extended as needed) along with keyframe data in A5/model.

Objectives

Name: Shawn Henderson
UserID: sbhender
Student ID: 00183318

__ 1: Environment generation responds to the displacement map

__ 2: Character keyframe animation is present, and responds to interaction

__ 3: Transparency effects blend with other elements properly

__ 4: Texture mapping has been used, with normal animation to create environment undulation.

__ 5: 3D Collisions are implemented, both static-dynamic and dynamic-dynamic

__ 6: Feedback is provided via an orthogonal HUD

__ 7: View Frustum Culling is implemented and can be seen via debugging mode

__ 8: Particle systems are present and are used effectively

__ 9: A sufficient Artificial Intelligence has been applied to computer controlled elements

__ 10: Sounds and music have been created, are present, and reflect game events

Declaration:
I have read the statements regarding cheating in the CS488/688 course handouts. I affirm with my signature that I have worked out my own solution to this assignment, and the code I am handing in is my own.

Signature: