Ethan Acklin

806 683-3694
eacklin@live.com

SKILLS

Languages
C/C++, Objective-C, HLSL, PHP, HTML

SDKs
DirectX, FMOD, WinSock, OpenGL ES, UIKit, Quartz 2D

Integrated Development Environment
Microsoft Visual Studio 2005,2008, 2010
Xcode 4

EDUCATION

DeVry University
Bachelor of Science in Game and Simulation Programming

PROJECTS

Brick Breaker App
2D iPhone application based on the original Brick Breaker game.

- Utilized personal 2D game engine for the base of the framework.
- Managed memory using an image cache.

Texas Hold'em Client
Multithreaded network based 3d game which utilized the Winsock and DirectX APIs.

- Dedicated two separate threads for receiving and sending data.
- Made use of critical sections to control synchronization of data.
- Optimizing data packets was handled through bit encoding.
- Used personal rendering engine interfaced with DirectX9.

Guns 'N' Grit
Designed and implemented an audio and physics core for a 3D racing game. Both cores were designed around the concepts of being plug and playable as well as being reusable, testable, and maintainable. These core architectures were achieved by using a modular design and other design patterns such as builder, singleton, façade, adapter, mediator, and chain of responsibility.

Audio Core

- Abstract class design using polymorphism provides for any 3rd party audio API, currently interfaced with FMOD.
- Support and full control for 2D or 3D sounds.
- Capable of playing long streams or short sounds.

Physics Core

- Resources are set up in a property centric design to be cache friendly.
- Realistic unconstrained motion was successfully simulated using a Runge-Kutta fourth order solver.
- Thrust, drag, and explosive forces allowed for a more dynamic and interactive experience.
- Spatial subdivision is managed through a hybrid setup of a grid and multiple octrees.
- Shape casting and line segment intersection allowed for predictive collision detection.
- The Separating Axes Theorem was used for nonintersection queries between spheres, axis aligned bounding boxes, oriented bounding boxes, and triangles.
- Implementing the Ritter Iterative Sphere algorithm produced a tight fitting bounding sphere from a mesh.
- Bounding boxes were obtained from a mesh through a brute force method of projecting all vertices.
- To handle terrain and other odd shapes, polysoups are used.