

Thomas Jones - Game Developer

www.tom-jones.net / thomas.antonio.jones@gmail.com

Education

M.S. in Computer Science - DigiPen Institute of Technology, Redmond, WA

Expected graduation April 2009, GPA: 3.95

- Thesis Topic: Temporally Coherent Patterns for Non-Photorealistic Rendering

B.S. in Computer Engineering - University of Florida, Gainesville, FL

Graduated May, 2005, GPA: 3.35: Cum Laude

A.A. in Engineering and Fine Arts - Santa Fe Community College, Gainesville, FL

Received July 2001

Skills

Programming Languages: C, C++, Managed C++, C#, XNA, .NET interop, Java, UnrealScript, HLSL

Graphics Libraries: Direct3D9, OpenGL, SDL

Graphics Techniques: Deferred shading, Phong lighting/shading, shadow mapping, hierarchical animation, per-pixel motion blur, particle effects, BSP trees, hierarchical occlusion mapping, non-photorealistic rendering, recursive ray-tracing, inverse kinematics

Miscellaneous: UnrealEditor 3.0 (including Kismet, Cascade, and Matinee), Source Engine SDK, Hammer Editor, Doxygen, Progress 4GL, Subversion SVN, Rational ClearCase source control, Microsoft Office

Employment History

Teaching Assistant *DigiPen Institute of Technology*, Help students and grade projects for the CS460 – Animation algorithms class dealing with interpolation techniques, hierarchical animation, motion along a curve, inverse kinematics. (Part-time, January 2009 – present)

Gameplay Programmer *Artificial Studios*, Designed and implemented game-play related objects and scripted levels for a PC and Xbox 360 title using UnrealEngine 3. (Full-time, June 2006 to August 2007)

- Released UE3 title *Monster Madness: Battle for Suburbia* in June 2007 for PC and Xbox 360.

Other positions: **Software Developer:** *Emdeon (WebMD)*, **Environmental Lab Technician:** *PPB Environmental Laboratories*, **Computer Lab Assistant:** *Santa Fe Community College Big Open Lab*.

Project Experience

Artificial Studios

Monster Madness: Battle for Suburbia

- UnrealEngine3-powered 4-player top-down action game released June 2007 for PC and Xbox 360. Group of up to 25.
- Implemented various enemy behaviors based on a design spec and assets, ranging from small mobs to entire bosses with scripted level interaction.
- Created custom Kismet objects for allowing unique level scripting actions as well as scripted several of the final levels.

- Implemented various weapons, items, and vehicle objects within the game.
- Networked various gameplay objects for use in multiplayer games.
- Became very familiar with using UnrealEditor and its Cascade, Kismet, Matinee, material, animation tree, prefab, and Archetype systems.

Unannounced UnrealEngine3-powered FPS shooter

- Estimated release late 2009, group of about 12 (formerly *Artificial Studios*, now *Ignition Entertainment*).
- Designed and implemented strongly object-oriented control, weapon, and player ability systems.
- Networked the control, weapon, and player ability systems for use in multiplayer games.

DigiPen

TEK - Sept. 2008 - April 2009, group of 4 programmers, 3 artists.

- Designer, lead programmer, *TEK*, a 3rd person melee fighting game that converts mouse movements into slash and thrust directions. Created deferred shading renderer in DirectX. Implemented a content pipeline for asset loading and object serialization. Designed and implemented a data-driven component-based architecture. Coded as significant portion of game logic, targeting, and camera control. Wrote major portions of the game design and technical design documents.

Drones - Jan.-April, 2008, group of 4 programmers.

- Technical Director and Designer, *Drones*, a non-combative puzzle-RTS. Implemented a content pipeline for asset loading and object serialization. Designed and implemented a data-driven component-based architecture. Wrote major portions of the game design and technical design documents. Conceived and implemented an intuitive scheme for controlling units in RTS.

Recursive Ray-Tracer - Sept.-Dec. 2008, solo project.

- Designed and implemented a recursive ray-tracer in C++ and managed C++. The ray-tracer compiles as a native C++ library and the front-end is a managed C++ WinForm, communicating via interop. Ray-tracing features included transmission, reflection, refraction, complex shapes using constructive solid geometry, and a pixel debugger pop-up window.

Wiimote gesture recognition via Neural Networks - April 2008, group of 2.

- Designed and coded a neural network that was used for recognizing gestures from a Nintendo Wiimote. Debug window shows neural net structure and colors the links based on edge weights.