IMGD 4000
Technical Game Development II
Introduction

Robert W. Lindeman
Associate Professor
Interactive Media & Game Development
Human Interaction in Virtual Environments (HIVE) Lab
Department of Computer Science
Worcester Polytechnic Institute
gogo@wpi.edu

What to Expect

- IMGD 3000
  - Mainly about the nuts and bolts of creating game-engine code
    - Game architecture, algorithms, data structures, mathematics
- IMGD 4000
  - Go deeper into some of these topics
  - Add new stuff, like networked multi-player, scene management, procedural content generation, AI, ...
- Presupposed background:
  - IMGD 3000: Technical Game Development I
  - CS 3733: Software Engineering
- Nice to have:
  - Computer Graphics

What to Expect (cont.)

- Today, there are many game engines available
  - Provide a starting point for game creation
  - Usually provide
    - Tools for importing content (*e.g.*, models, textures, *etc.*)
    - Scripting language to handle high-level control
    - Cross-platform support
- We want you to learn *what else* you need to make a modern game
  - Elements not provided by the engine
  - Elements to make your game unique

What to Expect (cont.)

- This course is about *game development* not Unity
  - But you will learn Unity
  - Focus on underlying methods
- This course is heavy on
  - Coding (C#)
  - Efficiency
    - Speed
    - Quality
    - Tech/Art intersection

Summary of Syllabus

- Lectures and in-class exercises
  - Designed to drive home concepts/projects
- Five Projects
  - Getting to know Unity (10%)
  - Basics + procedural content generation (10%)
  - Adding basic AI + interactables (10%)
  - Adding advanced AI (15%)
  - Adding multiplayer + polish (15%)
  - “Completeness” of final result (10%)
  - Extra credit too!
- Mid-term exam (15%)
- Final Exam (15%)

Resources for the Course

- Class Webpage
  - http://web.cs.wpi.edu/~gogo/courses/imgd4000/
- Discussion forum on at forums.gdc.wpi.edu
  - IMGD 4000 -> D-Term 2014
- Class email lists
- Video tutorials
  - www.design3.com
  - Requires a subscription ($20: Monthly, $60: Six months)
  - Required for Project 1
  - Useful for other projects

Resources for the Course (cont.)

- TA Jia Wang (wangjia@wpi.edu)
  - Rockstar Unity programmer!
- Lab sessions
  - Wednesdays, 2-3pm
  - IMGD Lab (FL-222)
  - Zoo Lab (FL-A21)
  - Sometimes joint with art folks
  - We'll let you know where each week
- Office hours
  - Please take advantage of these!

Unity Game Engine

- We will be using the Unity game engine for this course
  - You must do all your coding in C#
- Unity 4 is installed on all FL-222 & Zoo Lab machines
- You will use SVN for your projects
  - Jia will give a lab session on this

Projects

- Many phases to projects:
  - Understand/design/code/debug/test/eat/test some more
  - Encouraged to discuss approaches with others/in a group
  - On individual projects, work alone!
- Academic dishonesty (a.k.a., cheating):
  - Many reasons *not* to do it!
  - Immediate NR in the course
- Advice for doing well:
  1. Do the assigned tutorials (they are actually good!)
  2. Come to class
  3. Ask questions (class, office hours, WPI GDC discussions)
  4. Make sure you understand things before coding
  5. Don't share your code with others!

Term Project (Projects 2-5): Will be Interesting!

- Two-person tech teams, plus two-person art teams
- All teams will start with the same game design
- Art students will pitch themes
- You will focus mainly on technical aspects
  - You're not graded on your art results!
- Interim deadlines/presentations to show progress
- Presentations will be done the last day of this course, where you will show your stuff

But First…

- What is a ***game engine***?
- How does it work?

What is a Computer Game? **User Perspective**

- A goal (or set of goals)
  - Save the Princess (solve these puzzles first)
  - Score points (get power ups)
  - Finish first (unlock features)
- A set of rules governing game play
  - Turn taking, like RPGs
  - Reaction to events, like Tetris' falling blocks
  - Legal actions
- Visual (audio, *etc.*) content
- Control techniques
  - Button mappings

What is a Computer Game? **System Perspective**

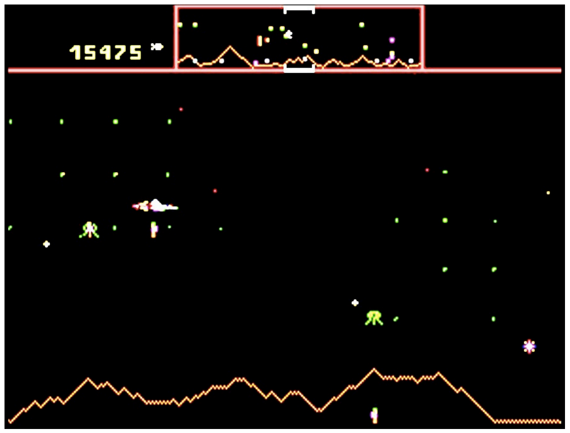
WPI

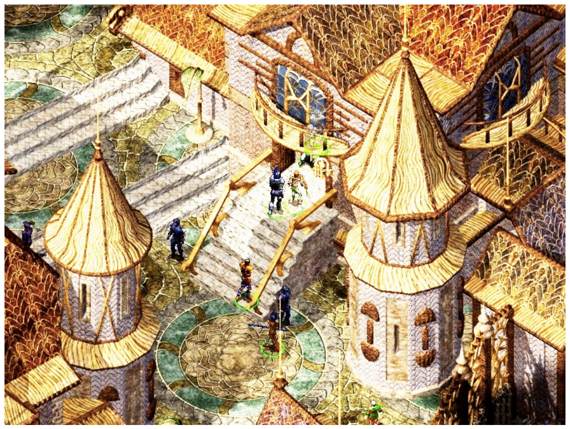
- A set of resources that are managed to support an entertainment (usually) application
- Graphical (audio, *etc.*) rendering
- A user interface
- Script handling
- Event processing
  - Time, collisions, *etc.*
- File I/O
- Asset-creation tools
  - Models, graphics, sound, *etc.*
- Optional
  - Networking
  - AI

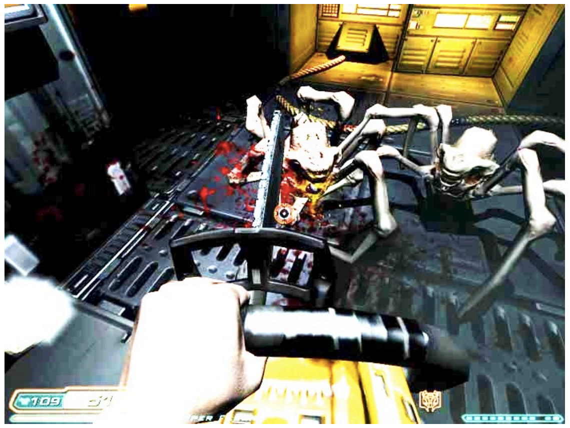
WPI

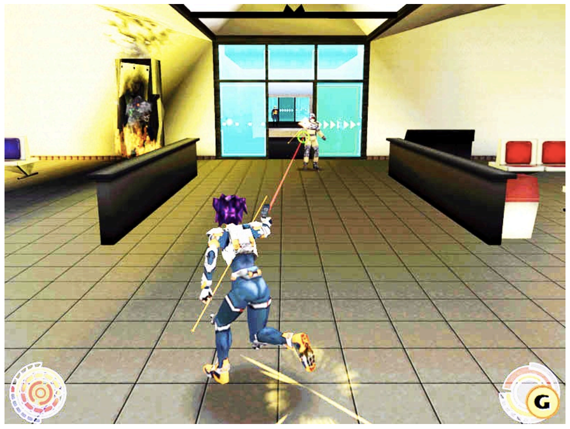
Types of Games

- 2D (Tetris)
- Side-scroller
- 3D isometric
- 1st-person view
- 3rd-person view
- Others too









Game Genres

- Genre defined:
  - A category of artistic composition, characterized by similarities in form, style, or subject matter.
- First-person Shooter (FPS)
- Real-time Strategy (RTS)
- Action
- Sports
- Simulation
- Stealth
- Puzzler
- Party

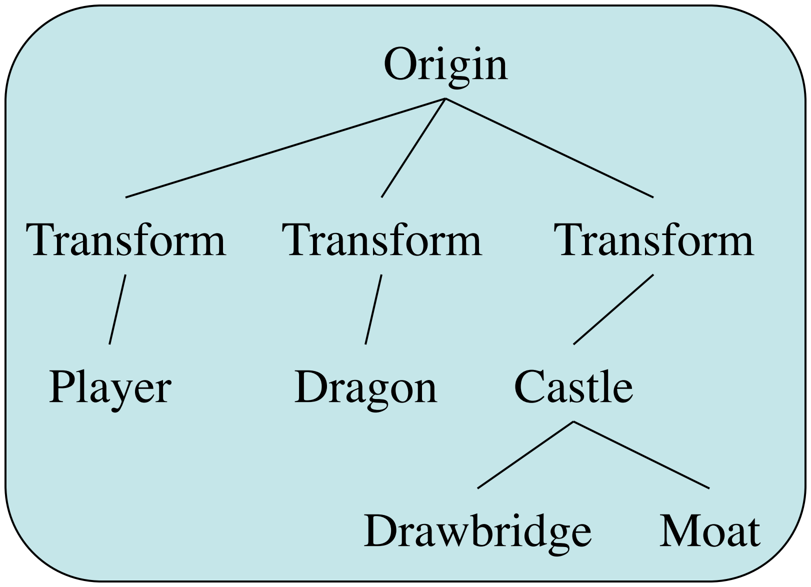
Elements of a 3D Game

- Game engine
- Scripting
- Graphical user interface
- Models
- Textures
- Sound
- Music
- Support infrastructure
  - Web site
  - Support forums
  - Admin tools
  - Database

WPI

Game Engine

- Scene graph
  - Representation of the world
  - Includes characters
- Timing is very important
  - Events
    - Time-based
    - Multi-player
  - Synchronization
- Database of objects
- Networking
  - Between server and clients
  - Between servers
  - Between clients



Game Engine (cont.)

- Core utilities
- Rendering system
- Physics
- Artificial intelligence
- Input management

Core Utilities

- Data structures
- Game-state management
- Timers
- Memory management
- Journaling services
- File logging
- Performance profiling tools
- Encryption/decryption

Scripting

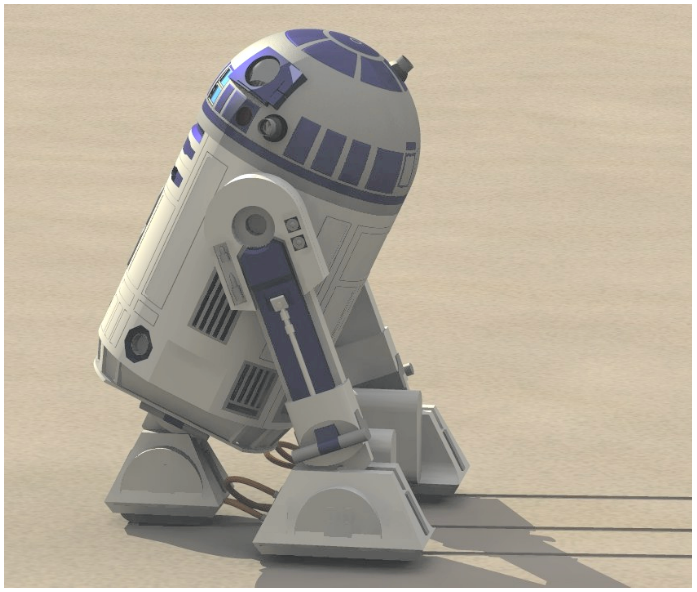
- Scripting languages provide easier path to building a game
  - Provides access to game-world objects (GWOs)
  - Allows most aspects of the game to be defined
  - Tie all parts of the game together
  - Leverage investment in engine development
  - Trade control for automation
- Sample scripting languages for games
  - Lua (www.lua.org)
  - Torque Script (www.garagegames.com)

Graphical User Interface

- Provides access to
  - Game menus (*e.g.*, save, load, boss)
  - Player status (*e.g.*, health, current speed)
  - Maps
  - Non-Player Character (NPC) dialog
  - Player-to-player chat

Models (Art Stuff)

- Objects are made from
  - Geometry (a.k.a., polygons)
  - Lighting
  - Textures
- Vertices and connectivity
  - Triangles
  - Triangle-strips
  - Meshes
  - Patches/surfaces



Texturing (Art Stuff)

- Created/manipulated using image processing software
  - Photoshop
  - Paint Shop Pro
- Mapped to geometry (models)
- Very powerful image enhancing techniques
  - Can be used for fake shadows, fake reflections, much more

Sound and Music

- One of the most-important elements of any experience is sound
- Sound effects
  - Make things more (hyper-) realistic
- Musical score
  - Sets the mood
  - Builds emotion
- Speech output
- Very important skill

Support Infrastructure

- Front-end for running games
  - Steam
- Web site
  - Promotion, log-in, etc.
- Support forums
  - Cheats, hints, discussion of new ideas
- Admin tools
  - User maintenance
  - Anti-cheating measures
- Database
  - Game-state maintenance

Our Focus

- We will focus on *tech stuff*
  - How to program a game from scratch
  - How to control characters (AI)
  - How to support networked multi-player
  - How to incorporate procedural content generation
- Less on content
  - Models
  - Textures
  - But, you will work with art students to wrangle their assets into your game