CSc-165
Computer Game Architecture

00 – Course Introduction

Dr. Scott Gordon
Computer Science
CSUS

"Design" vs. "Architecture"

- Game Design: how a game *looks* and *plays*
- Game Architecture: how a game is *built*

Note the difference from software engineering terms:

- In SE, "design" refers to the software *structure*
- In game engineering, "design" refers to the game (not the software that implements it)
- Game Design often involves storytellers, writers, artists, musicians, historians, etc.

This class is primarily about game *architecture*

Course Goals

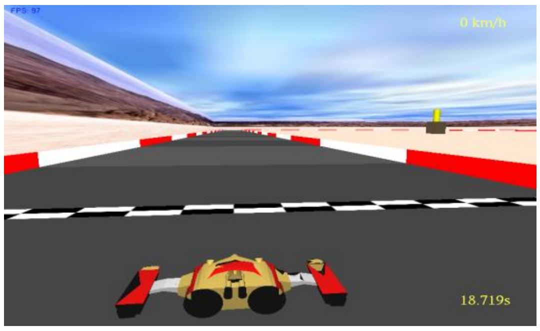
- The main goal of this course is to learn about the elements of game architecture
- This includes some hands-on experience building and modifying game engine internals (although rendering is taught in CSc-155)
- Although we will build our own games, building a great game is not the main goal of the course. Rather, it is the vehicle for learning game architecture. This is why we will use a very simple Java-based game engine that you will be able to add to and modify.
- That said, some great games will come out of the class!

Some game architecture topics:

- 3D virtual world construction and display
  (matrix transforms, terrain, skyboxes, textures, models, animation, lighting)
- Game Engine development
- Screen management
  (full-screen vs windowing, buffering, page-flipping, display rates)
- Player interfaces and controllers
  (render order, game console control, HUDs, object selection)
- Sound and music
  (linking sounds to events, spatial sound, platform independence)
- Artificial Intelligence (AI) in games
  (simulating intelligent behavior in NPCs, AI algorithms)
- Networking and massively-multiplayer games
  (client-server architecture, TCP vs UDP, network protocols)
- Physics worlds in games
- Scripting

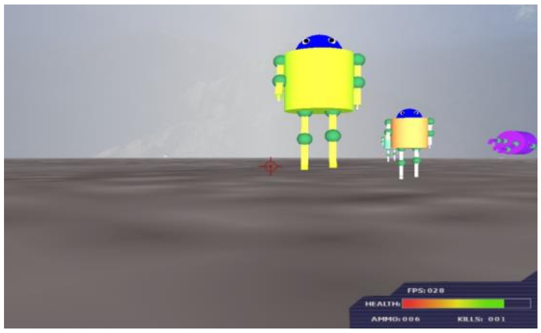
Example Games from past semesters

Racer



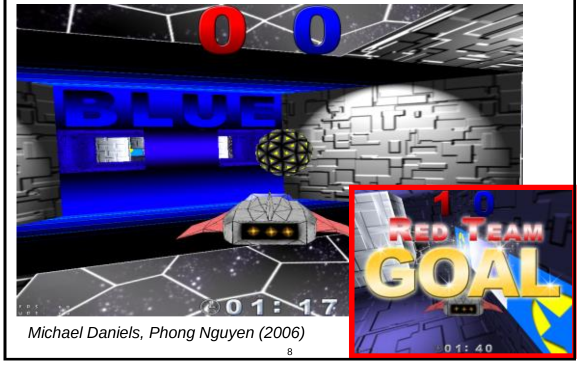
Joe Burks (2004)

Robot Overlords



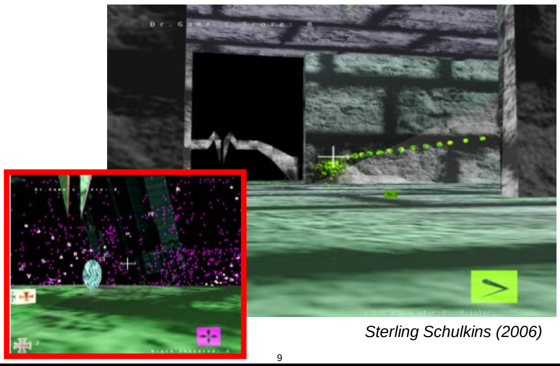
Joe Olivas, Luis Aguilar, Mike Outland (2004)

Starball



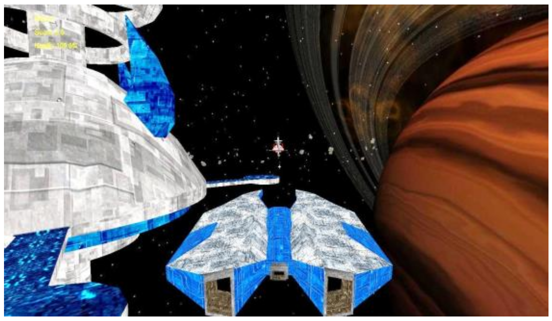
Michael Daniels, Phong Nguyen (2006)

Industria



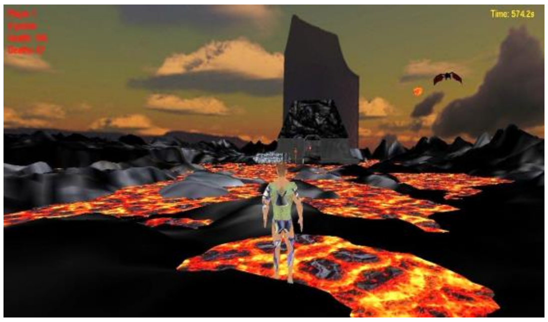
Sterling Schulkins (2006)

Base Raiders



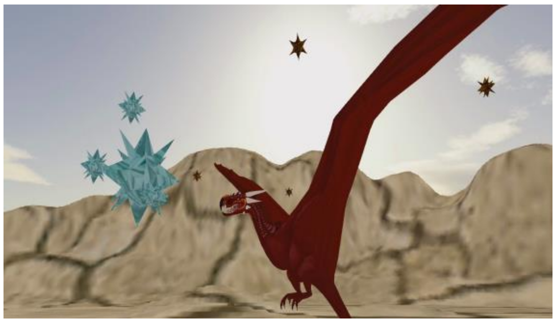
Ray Rivera, Tyler Creswell (2014)

Fire Fury



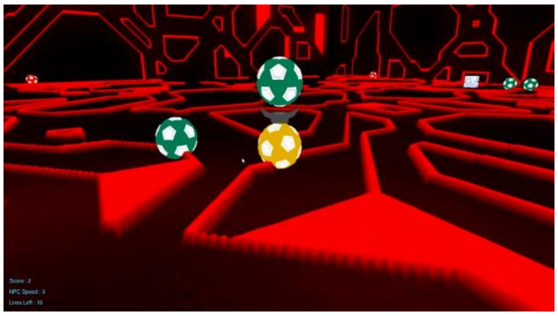
Sam Kerr, Justin Forrest (2014)

Hoard



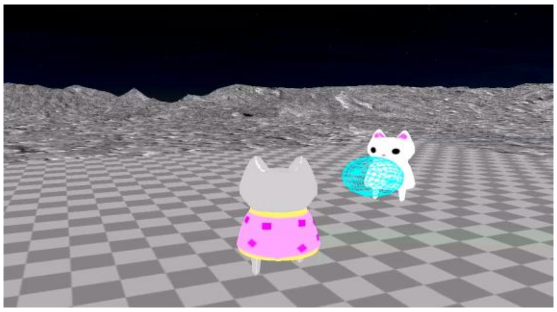
Alysha Straub (2015)

Glitch Ball



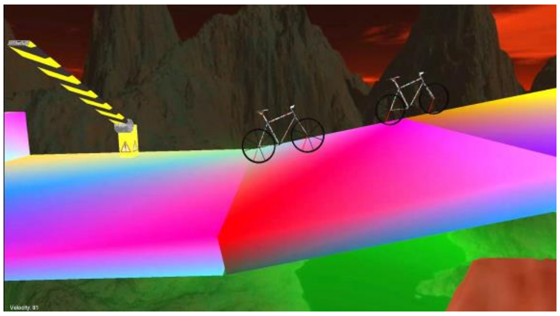
Nick Clayton, Travis Sutherland (2015)

Moon Cats



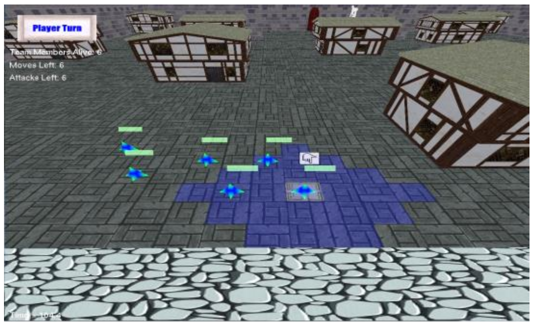
Stephen Ly (2015)

Bike Madness 16



Ben Botto, Bradley Dyer (2015)

Pillage

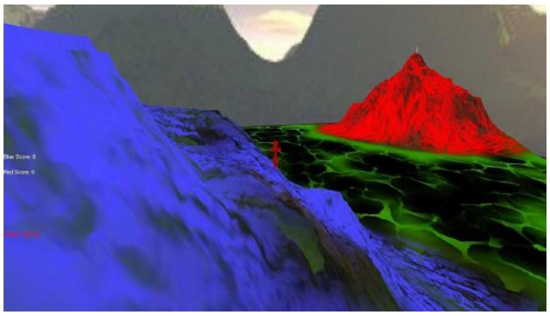


Kian Faroughi, Brandon Sherman (2015)

MoleSeeker

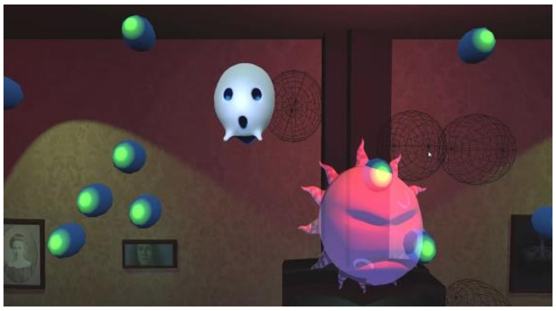
Mike Poku, Nietzu Kuan (2015)

Pixels vs. Texels



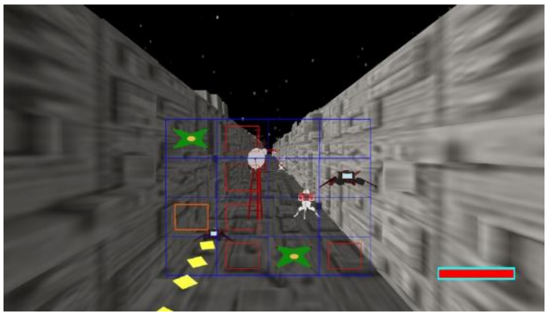
James Womack, Victor Zepeda (2016)

Haunted Mansion



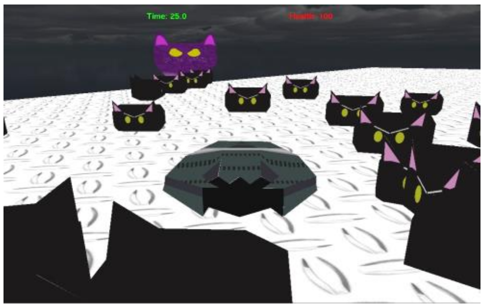
Dan Rogers (2016)

Trench Run



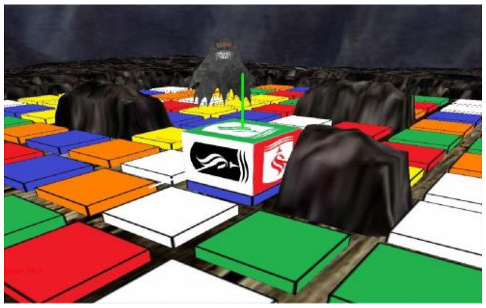
Matt Belcher, Jordan Jensen, Cody Malonee (2016)

Evil Space Cats from Space



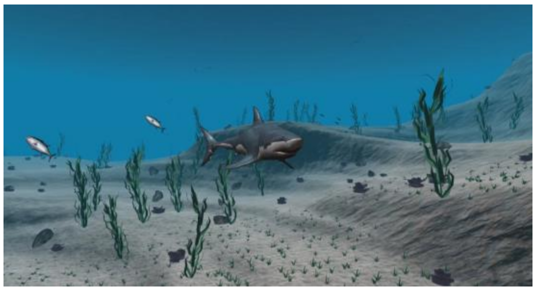
Greg Guzman (2017)

Cubix



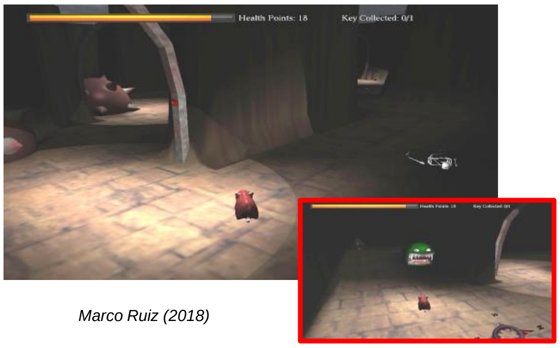
James Aldrich and Justin Tran (2017)

Bigger Fish



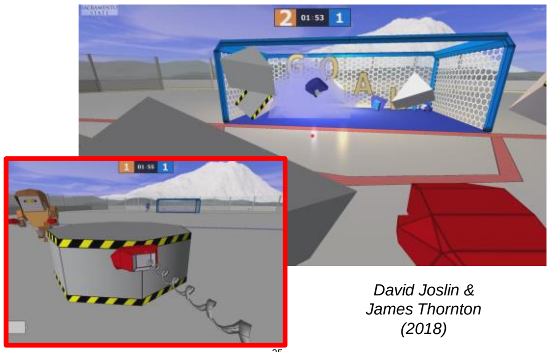
Chris Swenson (2018)

Horrific Maze



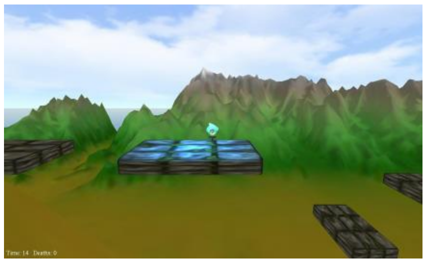
Marco Ruiz (2018)

Robo Hockey League



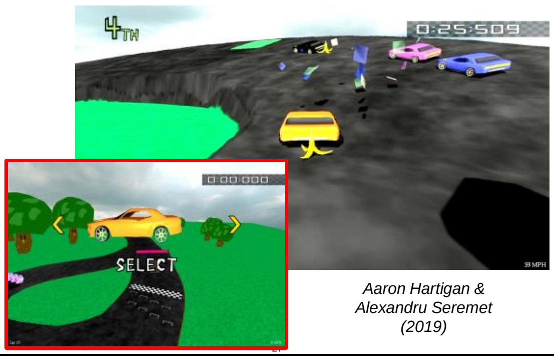
David Joslin &
James Thornton
(2018)

Platform Dynamics



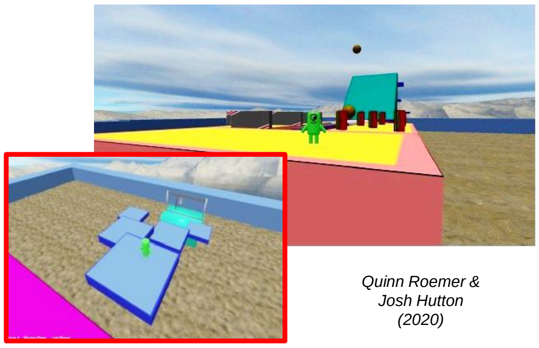
Alexey Zasorin &
Joshua Le (2019)

Luigi Cart



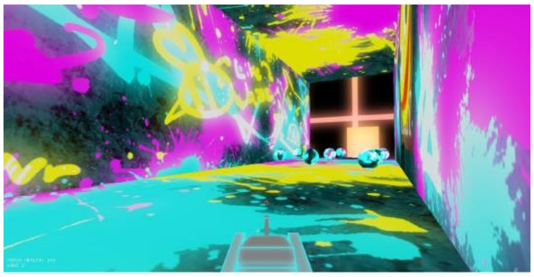
Aaron Hartigan &
Alexandru Seremet
(2019)

Gravity Guys



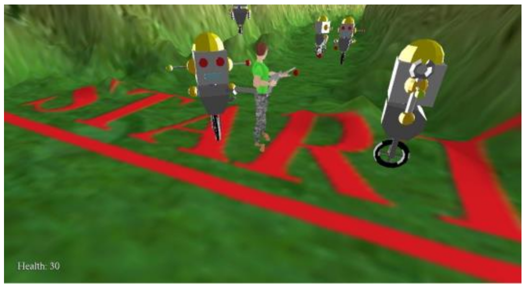
Quinn Roemer &
Josh Hutton
(2020)

Neonex



Micah Richardson
(2022)

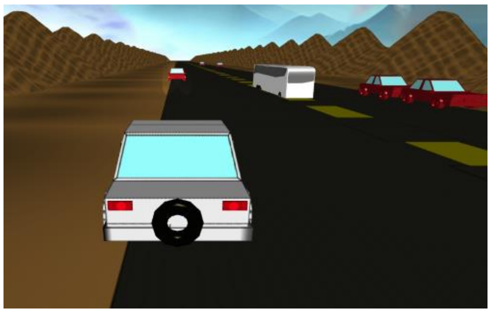
Robot Assault



Matthew Klaus
(2022)

Road Tage

Nicholas Burt and Girard Lin (2023)



Apocalpyse Arena

Prasad Prabhu and Jeffrey Tan(2023)

What will you build this semester?

Virtually all of the pieces of a game:

- Some game engine internals
- Camera and Node controllers
- 3D worlds and models, animations
- Handling input devices
- Physics and "physics worlds"
- 3D sound
- AI for non-player characters (NPCs)
- And much more!

oh, ok – let's talk briefly about "game design"

What goes into a game?

Gameplay

- What the players do when they are playing
- What makes a game "fun" or "interesting"

Art

- What players see (and hear) when they are playing
- Provides a game's "look and feel"

Technology

- How a game works
- Choosing and configuring an "engine"
- Hardware, devices, and system software

Gameplay: Genres

- Action (e.g., FPS)
- Adventure
- Role-playing (RPG)
- Real-time Strategy (RTS)
- Sports
- Simulation
- Management

Gameplay: Themes

- Wizards
- Alien Worlds
- Primitive Societies
- Medieval Conquest
- Earth in the Future
- Pre-existing concept
  e.g. Star Wars, NFL

Gameplay: Dimensionality

- Player motion
  - e.g., 0D, 1D, 2D or 3D
- Object and NPC motion
- View (camera) motion
- World dimensionality
  - e.g., ground, outer space

Gameplay: Activities

Examples:

- Exploration
- Combat
- Exploitation
- Physical dexterity
- Construction
- Destruction
- Story involvement
- Driving vehicles

Gameplay: Balance

Players must have “equally weighted” choices; game must “seem fair”

- Not too hard (or too easy)
- No “guaranteed winning strategy”

Requires *repeated, ongoing* play-testing

- Therefore, game must be built to allow changing relevant parameter values easily (e.g., scripting)

Gameplay: Balance (cont.)

Additional ways to achieve balance:

- Difficulty levels / level design
- “Catch-up” modes (variable NPC strength)
- Orthogonal differences in capabilities
- Avoid “brick walls”
- Avoid “free fall”
- Abstract/automate things that aren’t “fun”
  (but that can mean different things to different people)

Gameplay: Balance (cont.)

Avoid transitive strength relationships

- $A < B$ & $B < C$ → $A < C$
- Use non-transitive “Rock-Paper-Scissors” model

Avoid AI opponents that are

- Too strong
- Too fast
- Too smart

Power must be counter-balanced with weakness (e.g., powerful ammo, but limited amount)

Artistic Components

- Images
- Textures
- Lighting
- Level of Detail
- Sound & Music Composition

Virtually all games are Designed & Built by TEAMS

- Computer programmers
- Artists / Designers / Modelers
- Musicians / Foley artists
- Voiceover talent
- Businesspersons
- Domain experts
- Players

Virtually all games are built using an *Engine*

The engine handles:

- Low-level rendering
- Managing objects and models
- Device handling
- Math!
- Physics, sound, timing, etc.
- *things common to all games*

Our game engine - TAGE

- "Another Tiny Game Engine"
  - and yes, it is tiny!
- This is so you will write some game engine internals
- If you also take CSc-155, you will learn how to modify the *renderer*.

Computer Scientists are often hired by game companies to support their engine

And a final word of warning…

- This is *a heavy programming class!*
- It will take a *lot* of your time. *(and mine!)*

You just pressed here