Planning your Game (and semester)

CMSC425.01 Fall 2019
Instructional staff

- Prof. Roger Eastman
- Grad TAs:
  - Onur Kulaksizoglu
  - Tao Hu
- Office hours to start second week, to be posted
How do you plan for and build a game?

How will 425 help you do this?

Is this course right for you?
Background: What is the game industry like?

- Learn by reading new sites
  - gamastura.com
  - gamedev.net
  - hub.packpub.com
What's it like to be a game programmer?
(link: Cignition)

**Responsibilities**
- Write extensible and easily maintained game code using C# in the Unity game engine.
- Create asynchronous data-driven components capable of handling dynamic content received from a web server.
- Develop technical solutions for challenges faced in deploying a multi-platform game with limited processing and storage resources.

**Requirements**
- Mastery of software design fundamentals including object oriented and component based patterns, event driven systems, optimization, and debugging principals.
- Must have gone through a full commercial product cycle – from concept to shipping and post launch support in a role that included both architecting game systems and tracking down bugs.
- Must have a minimum of two years industry experience.
- Self-starter mentality to thrive in a startup environment, exploring a new problem space.
- Strong communication and collaboration skills.

Know Unity with C#
Handle Dynamic network content
Program Multiplatform, with limited resources
Collaborate/communicate in team
Who do you work with?

- Game design team
- Lead game designer
- Artistic director
- *Programmer*
- Level designer
- Tester
- Sound engineer
- Asset builder
Programmer's role

• Realize designer's vision

• Tweak gameplay and models

• Build supporting tools

• Extend game engine, build own

• (Toy Story 1 – tweak physics of spring to make look feel "right")
Do you want this job?

- Competitive career
- Crunch weeks common
- As good as your last game
- Unionization push!
- See: Blood, Sweat and Pixels by Jason Schreier
Game design vs. other software

• Process unpredictable
• Market fickle
• Expenses can be high ($100M or more)
• Develop at bleeding edge – keep advancing gameplay and appearance
Building an airplane in the air

• To build a game you must

• Design something unique, interesting and playable
• Fit into the cultural climate and gamer interests
• Advance tech features

• One approach: Sid Meier, Civilization designer (Baltimore)
• builds fun but crappy version by himself, then team rebuilds code from scratch
Must it be games?

• Design and navigate in 3D environments

• Simulate robots

• Prototype manufacturing floor
  • (Atlatl Software)

• Implement complex software rapidly
Game design team collaboration

• Need to be able communicate with all members of team
• Know a little bit of the entire process
• Understand each other's jobs

• Collaboration important this semester
  • In class collaborative activities
  • Collaborative game projects
Activity 1: organize your "game design" team

• At each table share

• Your names
• Interest in games
• What you want to get out of this class
• What role you'd most like in the industry (and on team)
Activity 2: Create a game! (Ice breaker)

• At each table

• Assembly your game packet (sheet, crayons, post-its, dice, pieces)

• Read the instructions

• Design a game in 10 minutes
  • Round robin – take turns making decisions
  • Make rapidly
Activity 2: Finish

• Put everything back in the bag
• Staple instructions and bag to board
• Label with your team #
Bkgrd: What's your game history?

- Mine
- Spacewar 1962 mainframe
- Asteroids 1970s
- Star Trek 1970s paper!
- Rogue 1980s text
Computer games

• Spectre on Mac 1990 - wireframe
• Decent 1994 – 8 bit full 3D FPS
• Starcraft 1998 –isometric 3D strategy
What's next?

• Mobile
• AR
• VR
• 3D sensors

Brain control
Going old school? Board games come back

• Board games

• Personal note: for me
• Computer games – play alone
• Board games - social
Activity 3: Your game history?

• At each table share

• What games have you played?
• Do you play now?
• Any experience with AR, VR, new types of games?
• What do you like?
Background: Programming a game

• What does it take to build a game?
Background: Building a game

• What software elements does it take?
  • 2D/3D rendering
    • Of environment, characters, objects, actions
    • Can be complex
  • Motion and navigation
    • Plan and execute motion from place to place
  • Physics
    • In "real time" games, simulate physics of object interaction
  • AI
    • Control motions and behaviors of non-player characters
• Databases and Networking
• Security
How put these elements together?

• Option 1: Write from scratch
  • Lots of work!
  • But, own, no payments

• Option 2: Assemble libraries (physics, rendering, modeling)
  • Less work, less payment
  • Less predictable!

• Option 3: Use game engine
  • Much less work
  • Good engine handles all for you
  • But not perfect, and generic- others have same tool
Supportive software: not for gameplay

• Create and manage assets
  • 3D modeling – build models of environment
    • Maya, Blender, Tinkercad, Pose
  • 2D imaging – create textures
    • Photoshop, GIMP, etc.
  • Asset management
    • Alienbrain

• Standard software engineering tools to test and maintain
  • Github, Buzilla, etc.

• Distribute final game
  • Steam, Apple App store, etc.
Activity 4a: Design a computer game

• At each table plan out a game for your team. Answer these questions (quickly!)

• What type of game? (platformer, FPS, RPG, etc. Multi-player?)
• What design choices?
  • Story
  • Environment
  • Characters
  • Gameplay
  • Visual look and feel
Activity 4b: Build a computer game

• At each table plan out a game for your team. Answer these questions (quickly!)

• What platform(s)?
• Any special hardware or peripherals needed?
• What software elements needed?
• Build from scratch or use engine? Which language or engine?
• What assets will you need? How will you make or get them?
CMSC425: Science and engineering of games

• How to build and tweak the software elements of games

• Topics
  • Game Engines
  • Geometric Programming and Data Structures
  • Modelling, and Animation
  • AI for Games
  • Motion Planning and Navigation
  • Networking and Online Games
  • Other
    • Physics, Audio, Particle systems and other procedural modeling, more
Workload and Syllabus

• Two introductory Unity projects
  • Learn to use a range of elements of Unity
• Final group project: Design and build your own game (your own team)
• Two midterm exams
• A limited number of major homeworks
• Minor in class and at home exercises

• Details at http://www.cs.umd.edu/class/spring2019/cmsc425/
• Schedule at Lectures link (has assignments, exams)
Readings

• Required: CMCS425 spring 2018 Lecture 1

• Suggested (and used in this lecture):
  • Blood, Sweat and Pixels by Jason Schreier
  • Indie Games: from Dream to Delivery, Don Daglow
  • *Game On!: Video Game History from Pong and Pac-Man to Mario, Minecraft, and More, Dustin Hansen
  
  • * I lived the history, didn't need a book!

• Next period: Game Engines and Unity. Look up Unity!
Summary

• After today you should be able:

1) Know and work with your classroom team
2) Describe the role of a game programmer in industry
3) Describe in general terms the members of a game design team
4) Describe in general terms how a game gets designed and built
5) List some of the software elements of a game
6) Explain why the game design process is often problematic