Engineering at a Games Company: What do we do?

Dan White
Technical Director
Pipeworks
October 30, 2019

The Role of Engineering at a Games Company

• Empower game designers and artists to realize their visions
  • Make tools and systems for designers and artists to use
• Engineering is still heavily involved in the creative process
  • Not always good…the creative process is brutal
  • Artists are trained to kill their children

What do you need to know?

• Typical CS Stuff – how to write large programs
  • Software development
  • Memory management, languages
  • Algorithms, Data Structures
  • User Interface (important)
  • Discrete Math
• Other stuff
  • Graphics
    • But not as much as you think
  • 3D Math
  • Simulation & Physics
  • Real-time networking
• Hopefully this class helps with the other stuff!
  • Useful not only for games, but machine vision, robotics, and so on

What tools do we use to make games?

• When Pipeworks started in 1999:
  • Blank hard drives
  • Visual Studio
  • 3ds Max SDK
• Now: Game Engines
  • Unity
  • Unreal
    • A few custom engines survive...
• Art is made with DCC tools
  • Maya, Blender, 3ds Max
  • Photoshop

Eventual Goal

• Eventually, artists and designers will be able to use engines to make games without engineers.
  • This is how it should be: You can make a document w/o engineering!
  • Don’t worry…this is decades off!
  • Many designers program, so the line is blurry
• The game engine will provide 95+% of the code needed to make the game
  • Again, typical: 95% of the code to display a web page is provided to you
• Our job is to provide what the game engine doesn’t

What do we actually Do?

• What the engines don’t do for free:
  • Fix performance problems
  • Simplified physics
  • Special Graphical Techniques
  • UI
  • AI
  • Procedural Content
  • Networking & back-end
  • Miscellaneous Yak shaving
  • Game specific code
Fix Performance

- Dev model: Artists add stuff until there is a problem then figure out why
- The goal is a consistent framerate
- Amortized speed doesn’t count
- Most important thing is to understand the rendering & update pipeline to find bottlenecks
- Solutions are often content changes, pre-calculation and so forth
- Threading when possible
- Rarely are perf problems fixed with just code changes
- Shaders are an exception
- Memory bandwidth problems can dominate

Simplified Physics

- Gameplay is hard to design. Physics is gameplay for “free”!
- Angry Birds is a demo for Box2D
- Free until it’s not – gameplay has to be predictable and understandable
- Many game engines have very sophisticated physics systems
  - The math is crazy
  - Check out Bullet Physics
- Engineering needed for
  - Optimizations
  - Fractures
  - Predictable behavior
  - Tires/Cloth/Soft bodies
- Many games do better without a complex physics simulation
  - E.g. Roller Coasters

Special Graphical Techniques

- Most shaders can be made by artists
- DCC tools make graphics easy
- Writing shaders is now a technical art position
- See Brutal Legend Ink

Curved World in Animal Crossing

This is done with a vertex shader the world is flat!

Other special techniques...

- Borderlands 3 – Cell Shading
  - Monument Valley – perspective rendering tricks

AI

- A famously vague term
- For games we usually want:
  - Satisfying opponents
  - Relievable NPC’s
  - Optimality not required (or even desirable)
- Usually bespoke and rule-based
  - Harder than you might think
  - Have to know rules in detail
  - Check out Steering Behaviors For Autonomous Characters
  - We have been trying to make autonomous vehicles long before it was fashionable. Good luck
- A lot of interest in reinforcement learning techniques.
UI

- User interface is important
- Often mixes with 3d in the world
- Rendering is done by the 3d pipeline
  - Using faster than raster methods
- Typical Pipeline:
  - Screen mock-ups made by designers
  - Pretty is added by artists
  - Functionality is from engineering
- Lots of color, and animation and VFX
- Madden: 500 screens

Procedural Content

- Stuff that artists and designers don’t make
- Allows replayability at low-cost
- Avatar systems
  - E.g Character Creation
- User created structures
  - E.g. building in Fortnite
- Foliage
- Crowd and background characters
- Terrain
  - The world in Minecraft or Terraria
- Very game-specific

Networking and Back End

- Managing & debugging a distributed state machine...hard
- Need to hide latency
  - TCP is not good
- Typically use UDP with some sort of reliability layer – check out Enet
- Ration bandwidth
- Error handling
  - Everything that can go wrong, will...a lot...and users will make it worse
- Interface with databases
- Predictable performance can be hard

Yak Shaving

- Engineering owns the build/deploy tool chain
  - Jenkins/Ci etc.
  - Source control, which is notably difficult for Games
  - Git model does not work as well (but LFS helps)
- Satisfy Console and Platform requirements
  - Far more rigorous than the App Store
- Every software business has this stuff
- We need strong programmers

Game Code

- Camera & Control
- Game rules
- Character animation
- And so on...

Game Code Examples...

Roller Coasters
Fighting Game Camera
Roller Coaster Games

- Tracks are 3D splines
- Splines are edited in game
- Train physics are simple 1-D models
- Physics engine used for cars that come off the track
- Have to procedurally create track meshes

Fighting Game Camera

- Frame the action
- Follow the characters
- Nice transitions
- Godzilla: Destroy All Monsters Melee code from 2001 is 500 lines
- Includes tracking and a small state machine

Underlying Skills diff vs. Typical CS

- 3D Math
- Matrices
- Simple physics
- Blending
  - Nature is smooth
- Mesh Manipulation
- Robustness

Robustness: Floating Point is the Devil

- What does this return?
  - Does it even return?

```cpp
float add_forever()
{
    float t = 0;
    while (1)
    {
        float next = t + 1.f / 30;
        if (next == t)
            break;
        t = next;
    }
    return t;
}
```

Answer

1048576.00 = 2^20 = 2^25 / 2^5

- If you update your simulation time this way, time stops after ~12 days
- Most games & graphics software runs on 32-bit float
- A big issues for flight sims and large worlds
- Safety in double is illusory anyway

Questions?