LECTURE 11
Announcements
Final 2 Feedback

• Include relevant INSTRUCTIONS and READMES!
• It is very hard to grade without an indication of where to look
• This can lead to undeserved incompletes
Final 3 and 4

- You should setup a time to meet with your mentor soon, before Friday
- Make Final 3 and 4 rubrics for gameplay
- Write your game
- $$Profit$$
Deadline Approaching

• Course policy: you must turn in a working version of the engine for every project
• Deadline for incomplete projects is December 17
• Same day as Final V
• Make sure to email the TA staff when you re-turn in!
Announcements

QUESTIONS?
LECTURE 11
AI in Games
Considerations in Game AI

- Available Information
- Randomness
- Timing
- Coordination
- Symmetry
- Cooperation
Information

• **Perfect Information**
  - Know the full state of the game at all times
    • Chess
    • Go

• **Incomplete Information**
  - Only know part of the game state initially
    • Stratego
    • Fog of War
  - May need to model what is currently known
Randomness

- The initial state or outcomes of actions are often not deterministic
- Can make decision making much more complicated
Timing

• Real Time vs Turn Based
  – If game is real time, then decisions must be made relatively quickly

• Simultaneous Action
  – If players make simultaneous decisions, then should factor in what other players could do into decision making
Coordination

• **Single Actor**
  – Every Entity makes decisions independently
  – Simpler than alternatives, but less powerful
  – Might make sense for puzzle games

• **Global Planning**
  – Single AI plans out actions of each individual entity
  – Good for strategy games

• **Inter-entity Communication**
  – No global plan, but nearby entities work together
  – Can allow for more realistic behavior in most games
Symmetry

• Often the AI is fundamentally playing the same game as the player
  – Same possible actions, often same goal
  – Most Fighting Games
  – Most RTS
  – How does the AI and game need to be designed so that the AI can use the same systems as the player?

• Sometimes the AI is playing by completely different rules
  – Many Simulation Games
  – AI Director in Left 4 Dead
  – How might players take advantage of these separate sets of rules? Can the AI be abused?
Cooperation

• Sometimes AI have to cooperate with players or other AI
  – Does the AI communicate with the player, and if so how?
  – How do you keep the AI at the right amount of helpfulness?
AI in Games
SEARCH
Game Tree

- Explicit representation of the decisions the agent can make from the current state along with the possible outcomes.
- State change
- Opponent moves
- Moves of “nature”
Game Tree

- Computation relatively easy
- Expand to bottom
- Leaves have values
- “max” up our decisions
- “min” up opponent’s
- “expected value” up nature
- Take best move at root
Tree Complications

- Tree gets big fast
- Exponential in depth
- Tree can be deep
- Won’t reach leaves
- Can be complicated by incomplete information, or randomness
Single-agent decisions

- Find the cheapest sequence.
- Can build a full game tree.
- End up visiting all possible nodes.
A*

• With additional information, can prune and avoid expanding some nodes.

• Admissible heuristic:
  • Optimistic estimate of cost to goal

• Always expand node with smallest cost to reach plus estimate to complete.

• Optimal paths using information optimally!
Admissible Heuristics

- Straightline distance
- Relax constraints
- Background knowledge
- MST vs traveling salesman
Pruning Game Trees

- Although we want to exact value at the top of the game tree, we might only need accuracy within a range elsewhere.
- Return:
  - Beta if true value is beta or above (more than enough),
  - Alpha if true value is alpha or below (not enough),
  - exact value otherwise.
**Alpha-Beta**

- **Max node:**
  - First child: ask for alpha,beta
  - If bigger than beta, return beta.
  - If bigger than alpha, increase alpha.

- **Min node:**
  - First child: ask for alpha,beta
  - If smaller than alpha, return alpha.
  - If smaller than beta, decrease beta.

- Root: negative to positive infinity.
- Not optimal in terms of effort!
Evaluation Function

- Even with pruning, can’t reach leaf.
- Use evaluation function to guess value.

- $P = 100$
- $N = 320$
- $B = 330$
- $R = 500$
- $Q = 900$
- $K = 20000$

- Human expertise.
- Machine Learning can help.
Monte Carlo Tree Search

• A new approach to search problems you might find useful!
• Applications and strengths and weaknesses compared to traditional search methods.
MTCS vs. alpha-beta

- Consider sequences at full depth, but do not consider all possible actions. Heuristic procedure to select promising lines of search.
- alpha-beta: Wide but shallow, heuristic to fill in value below.
- MCTS: Deep but narrow, heuristic to select actions along the way.
BALANCING GAME AI
General AI vs Game AI

General AI
- Find the best solution to a problem
- Learn from the past/training
- Imitate human decision making

Game AI
- Provide opponents/obstacles
- Give reasonable and predictable [default] behavior
- Make the game fun
Balancing AI

• The optimal AI is generally too good at most parts of a game
  – Computer programs are now good enough to beat the best human players at Chess and Go
  – Too good at execution (Aim in shooters, combos in fighting games, etc.)
• You can make an AI that is not optimal
  – But how do you get the balance right?
Balancing AI

• You can affect the balance by removing symmetry
  – Give AI characters a larger spread to aim in shooters if they are too good
  – Make AI better at an RTS by giving it more resources, or give it perfect information in an incomplete information game
  – Can make it easier to add difficulty levels by changing how much of an advantage/disadvantage the AI gets
OTHER AI TECHNIQUES
Machine Learning

- Cool idea, not very applicable to game AI
- Could pretrain parameters to be used in other AI techniques
- Could try to have AI improve from past sessions
  - Amiibos do something like this in Super Smash Bros
Natural Language Processing

- Can be useful in specific games
  - Games with text input
  - May want to use training data with game-specific language
AI in Games

Annual AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment

- **AI in Game Design**: AI as a source of novel game mechanics and genres
- **AI-Based Production and Authoring Tools**: Behavior-building, design frameworks, telemetry-supported game design, content authoring support, scripting, sketch-based authoring, automated playtesting
- **AI Techniques for Games**: Planning, reinforcement learning, search, neural networks, Bayesian models, evolutionary algorithms, case-based reasoning, constraint programming, utility-based approaches, animation, camera control, tactical/strategic decision making, terrain analysis, opponent modeling, dynamic difficulty adjustment, spatial decompositions, path planning
- **AI Storytelling**: Interactive drama, story generation, character development
- **Autonomous Characters, NPCs, and Virtual Humans**: Personality, emotion, believability, natural language processing, cognitive modeling, crowd simulation, social robotics
- **Procedural Content Generation**: Level generation, progression design, behavior adaptation
- **Commercial AI Implementations**: Case studies, implementation analysis, comparative evaluations
- **AI in Novel Entertainment Applications**: Entertainment robotics, virtual/mixed reality, mobile device games, geo-location based games, games for human-computation
- **Computational Creativity and Generative Art**: Painting, poetry, story, humor, music
- **AI in Games for Impact**: Training, education, intelligent tutoring, games for health, gamification
AI in Games

QUESTIONS?
UI principles

• Does it tell me what I need to know?
League of Legends
League of Legends
UI principles

• Does it tell me what I need to know?
• Is it clear how everything works?
Civilization V
Tooltips
FTL
Tutorials

This is your equipment screen. You can see more detailed information about your weapons, drones, or augments by mousing over them.

Click and drag your new Halberd Beam to replace either of your current weapons, then hit ACCEPT.

Halberd Beam
Slow but reliably powerful standard beam weapon

Note: Beam damage is reduced by one for every shield it passes through, which allows for partial shield piercing.

Required Power: 3
Charge Time: 17 seconds
Damage Per Room Hit: 2
Break Chance: Low
UI principles

• Does it tell me what I need to know?
• Is it clear how everything works?
• Is it easy to use?
League of Legends
Diegetic UI
Meta UI
Meta UI
Morals

• Be creative
• Make sure the appropriate information is conveyed
• Get lots of feedback
• Don’t be afraid to iterate
• Take CS130!
QUESTIONS?
Content Management

- Designer can make content
- Iterate faster and without recompiling
- Potentially reload content at runtime
Representing Behavior

• How to encode AI or map-specific gameplay?
  – Could use Entity I/O
  – Could use behavior trees, FSM, other structures

• Or, we could use code to express complex behavior
Two languages: worth it?

- One language would be simpler
  - No marshaling needed
  - No API needed
- Could just use Python
- Could just use Java
  - Can dynamically load Java source

- Core engine needs to be fast
  - Collision detection
  - Physics
  - Rendering
- Scripts need to be sandboxed
  - Security
  - Convenience
Complementary Features

C++

- Statically typed
  - Maintainable

- Compiled
  - Long build times

- Direct control of memory
  - Good for CPU-intensive code

Scripting language

- Dynamically typed
  - Flexible

- Interpreted
  - Slower at runtime

- Protection from machine details
  - No memory corruption
Lots of choices

- **General-purpose**
  - Python (Jython)
  - Javascript (Rhino)
  - Ruby (JRuby)

- **Designed for embedding**
  - Lua

- **Designed for engines**
  - UnrealScript
  - Game Maker Language

- **Misc**
  - Pawn, TinyScheme, AngelScript, Squirrel, GameMonkey, etc
Jython

- Create an interpreter
- Convert Java objects into Python objects (marshaling)
  - Can also be automatic using reflection
- Run Python scripts

```java
PythonInterpreter py = new PythonInterpreter();
py.set("x", new PyInteger(42));
py.exec("y = x / 6");
PyObject y = py.get("y");
py.execfile("myscript.py");
```
Lua

- Small learning curve
- Almost everything is a (hash)table
- First-class functions
- Light and efficient
- Relatively simple C integration
- Most popular game development scripting language

```lua
players = { "Peter", "Paul", "Mary" }

scores = { Mary = 10, Paul = 7, Peter = 8 }

table.sort(players, function (i, j)
    return scores[i] > scores[j]
end)
```
UnrealScript

- Used in the Unreal Engine (proprietary)
- Static types
- Networking concepts
  - Replication
  - Simulation
- Editor integration
- Function calls can last several frames
  - Each Entity gets its own logical thread (not OS thread)
  - Much easier than callbacks

```cpp
class FlashLight extends SpotlightMovable;
var FlashLight MyFlashLight;

var repnotify bool bIsOn;
simulated function Toggle()
{
    PlaySound(FlashLightSound);
    // ...
}

d() Color CaseColor;

reliable server function Pulse()
{
    Toggle();
    sleep(0.5);
    Toggle();
}
```
YourOwnLanguage

• Take cs173
  – Learn about language implementation
  – Learn about language design tradeoffs
  – Mostly ignores performance
• Take a compilers course
  – Focus more on performance
  – Implement optimizations
  – It runs in the fall
• Jump in blind!
Organization

- It’s inconvenient to keep track of a ton of script files
  - Consider integrating script editing into the engine
- Shouldn’t have to hard code which script file to use
  - This you can easily do with 195n editor
  - …and should be able to do with any level editor
Language Interaction

• API design: how do the scripting language and high level language communicate?
• Some considerations:
  – Security: script should be sandboxed if users can add scripts
  – Utility: script should be useful
  – Abstraction: script shouldn’t have to do legwork
• Designing APIs is hard!
  – Could use some neat reflection tricks to wrap all Java methods
  – Could use a library to simplify the entire process
Embedded Scripting

QUESTIONS?
What kinds of games?

• High replayability
• CPU players aren’t enough
• Genres?
  – Fighting, FPS, MOBA
• Less likely?
  – Story-based games (Wolf Among Us, Undertale, Earthbound)
Local Multiplayer

- Controllers are (almost always) necessary!
- Other options exist, but at a cost (many keyboards/mice)
- Why no local LoL or Starcraft?
Online Multiplayer

- Mostly keyboard & mouse
- Better control (esp. with mouse)
- More control options (with keyboard)
- Exception: fighting games
This Class?

- Keyboard & Mouse is your only option...
- Probably using networking
- What about local multiplayer?
  - Watch out for key ghosting
Multiplayer Game Design

ASYMMETRY
What is it?

• Players having different abilities/gameplay experiences than other players

Why have it?

– Variation: less repetitive gameplay
– Strategy: improve game meta (e.g. smash bros)
– Theming: fitting the game’s story/environment
  • Monster vs. hunters, Attackers vs. Defenders, etc.
Asymmetry

Nonphysical

Physical
Asymmetry (cont.)

Nonphysical
- Character/Class Selection
- Environment attributes
- Randomization
- For turn-based: who goes first?

Physical
- Major differences in gameplay
- Wii-U gamepad
- Verbal: Keep Talking and Nobody Explodes
Perfect Symmetry...?
Multiplayer Game Design

USER EXPERIENCE
Local Multiplayer

- Primary issue: fitting everyone on a single screen
- Two primary options
  - Splitscreen: FPS & Racing
  - Same screen: Fighting & Board Games
  - Platformers: both?

Robots Are People Too (2012)
Online Multiplayer

• Primary issue: finding players to play against
• Ratio of play time to wait time
• Players are impatient
Just make wait times short!

...right?

NOPE.

Make them feel shorter
Mitigating Wait Times

- Make a lobby
- Allow any single player mode while waiting
- Not like Super Smash Bros. Brawl
This Class?

• Most likely: a basic text input, user types IP
• Easy: playground/lobby while waiting
• Hard: lobby with other players
Gameplay is Important

- Implement most of your game logic this week
- Playtest early and often
- Use the feedback you’re getting
NO PLAYTESTING
You should be getting feedback from people outside the class