CS 354R: Computer Game Technology

Introduction to Game AI
Fall 2019
What is AI?

- AI is the control of every non-human entity in a game
  - The other cars in a car game
  - The opponents and monsters in a shooter
  - Your units, your enemy’s units and your enemy in a RTS game
- But, typically does not refer to passive things that just react to the player and never initiate action
  - That’s physics or game logic
  - e.g blocks in Tetris are not AI, nor is the ball in the game you are doing, nor is a flag blowing in the wind
  - It’s a somewhat arbitrary distinction…
AI in the Game Loop

- AI is updated as part of the game loop, after user input, and before rendering
- There are issues here:
  - Which AI goes first?
  - Does the AI run on every frame?
  - Is the AI synchronized?
AI in the Game Loop

- Consider how an AI system might need to interact with other game systems

- https://www.youtube.com/watch?v=7ESipcQunHc

- https://www.youtube.com/watch?v=8x9xoxB1DfI

- https://www.youtube.com/watch?v=6402TvQMPkU
AI and Animation

• How should AI and animation relate?
  • Scenario 1: The AI issues an order (move from A to B), and the animation system controls character accordingly
  • Scenario 2: The AI controls everything including which animation clip to play
• Controls depend on the AI and animation systems
  • Is the animation system based on move trees (motion capture), physics, or something else?
  • Does the AI handle collision avoidance? Does it do detailed planning?
AI Update Step

- Sensing
  - Determine state of the world
  - May be very simple - state changes all come by message
  - Or complex - figure out what is visible, where your team is, etc
- Thinking
  - Decide what to do
- Acting
  - Execute on decision
  - Notify animation and world state
AI by Polling

- The AI gets called at a fixed rate
- Sensing: agent looks to see what has changed in the world
  - Queries what it can see
  - Checks if its current animation has completed
- Thinking: agent decides on an action
- Acting: agent acts

- Why is this generally inefficient?
Event Driven AI

• Event-driven AI responds to changes in the world
  • Events sent by message just like the user interface
• Example messages:
  • A certain amount of time has passed, so update yourself
  • You hear a sound
  • Someone has entered your field of view
AI Techniques

- Basic problem: Given the state of the world, what should I do?
- A wide range of techniques used in games:
  - Finite state machines, decision trees, rule-based systems, neural networks, fuzzy logic, behavior trees
- A wider range of solutions in the academic world:
  - Complex planning systems, logic programming, genetic algorithms, Bayes-nets
  - Typically, too slow for games but becoming more common
Goals of Game AI

• Desirable Characteristics:
  • Goal driven - the AI decides what it should do, and figures out how to do it
  • Reactive - the AI responds to changes in the world
  • Knowledge intensive - the AI knows a lot about the world, and embodies knowledge in its own behavior
  • Characteristic - Embodies a believable, consistent character
  • Fast and easy development (designer-controlled)
  • Low CPU and memory usage
• Of course, these conflict in almost every way…
Two Measures of Complexity

- **Complexity of Execution**
  - How fast does it run when knowledge is added?
  - How much memory is used when knowledge is added?
  - Determines the run-time cost of the AI
- **Complexity of Specification**
  - How hard is it to write the code?
  - As knowledge is added, how much more code is written?
  - Determines the development cost, and risk
Expressiveness

- What behaviors can be easily defined, or defined at all?
- Propositional logic:
  - Statements about specific objects in the world (no variables)
  - Jim is in room7, Jim has the rocket launcher, the rocket launcher does splash damage
  - Go to room8 if you are in room7 through door14
- Predicate Logic:
  - Allows general statements (using variables)
  - All rooms have doors
  - All splash damage weapons can be used around corners
  - All rocket launchers do splash damage
  - Go to a room connected to the current room
Finite State Machines (FSMs)

- A set of the agent’s states
- Transitions between states triggered by a change in the world
- Represented as a directed graph (edges labeled with the transition events)
- Ubiquitous in computer game AI
- You might have seen them in formal language theory or compilers
Consider the bot AI of an arena shooter (e.g. Quake). What do we need in our FSM to capture some of its desired base behaviors?
Quake Bot Example

• Types of behavior to capture:
  • Wander randomly if no sight or sound of an enemy
  • When enemy is seen, attack
  • When enemy is heard, chase
  • When death, respawn
  • When health is low and enemy is seen, retreat

• Extensions:
  • When power-ups are seen, collect

(from John Laird and Mike van Lent’s GDC tutorial)
Example FSM

- **States:**
  - E: enemy in sight
  - S: sound audible
  - D: dead

- **Events:**
  - E: see an enemy
  - S: hear a sound
  - D: die

- **Action performed:**
  - On each transition
  - On each update in some states (e.g. attack)
Example FSM Problem

- **States:**
  - E: enemy in sight
  - S: sound audible
  - D: dead

- **Events:**
  - E: see an enemy
  - S: hear a sound
  - D: die

**Problem:** Can’t go directly from attack to chase. Why not?
Better Example FSM

- **States:**
  - E: enemy in sight
  - S: sound audible
  - D: dead

- **Events:**
  - E: see an enemy
  - S: hear a sound
  - D: die

- Extra state to recall whether or not heard a sound while attacking
Example FSM with Retreat

- **States:**
  - E: enemy in sight
  - S: sound audible
  - D: dead
  - L: Low health

- **Worst case:** Each extra state variable can add $2^n$ extra states
  - $n = \text{number of existing states}$
Hierarchical FSMs

- What if there is no simple action for a state?
- Expand a state into its own FSM, explaining what to do
- Some events move you along the same level in the hierarchy, some move you up a level
- When entering a state, choose a state for its child in the hierarchy
  - Set a default, and always go to that
  - Or, random choice
  - Depends on the nature of the behavior!
Hierarchical FSM Example

- Note: This is not a complete FSM
  - All links between top level states still exist
  - Need more states for wander

```
Start
Turn Right
Go through Door
Pick up Powerup
Wander
```

```
Attack
~E
E
~S
S
D
~E
```

```
Chase
```

```
Spawn
```

```
~E
E
~S
S
D
~E
```
Non-Deterministic Hierarchical FSM (Markov Model)

- Adds variety to actions
- Have multiple transitions for the same event
- Label each with a probability that it will be taken
- Randomly choose a transition at run-time
- Markov Model: New state only depends on the previous state

```
Attack

Start

Approach

Aim & Slide Right & Shoot

Aim & Slide Left & Shoot

Aim & Jump & Shoot

.3

.3

.4

.3

.4

```

- .3
- .3
- .4
- .3
- .4
“Efficient” Implementation

- Compile into an array of state-name, event
  \[\text{state-name}_{i+1} := \text{array}[\text{state-name}_i, \text{event}]\]
- Switch on state-name to call execution logic
- Hierarchical
  - Create array for every FSM
  - Have stack of states
    - Classify events according to stack
    - Update state which is sensitive to current event
- Markov: Have array of possible transitions for every (state-name, event) pair, and choose one at random
FSM Advantages

- Very fast – one array access
- Expressive enough for simple behaviors or characters that are intended to be “dumb”
- Can be compiled into compact data structure
  - Dynamic memory: current state
  - Static memory: state diagram – array implementation
- Can create tools for non-programmers to build behavior
- Non-deterministic FSM makes behavior unpredictable
FSM Disadvantages

- Number of states can grow very fast
  - Exponentially with number of events: \( s = 2^e \)
- Number of arcs can grow even faster: \( a = s^2 \)
- Propositional representation
  - Difficult to put in “pick up the better powerup”, “attack the closest enemy”
  - Expensive to count: Wait until the third time I see enemy, then attack
    - Need extra events: First time seen, second time seen, and extra states to take care of counting