Video Game AI: Lecture 3
Decision Trees (brief) and Finite State Machines

Nathan Sturtevant
COMP 3705

Lecture Outline

• Was going to start pathfinding
  • But, some details today that will really help thinking about your HW assignment

• Any problems with the HW?

Meet with Brian Robbins - April 5 (Contact Susan)

DU CS alumnus Brian Robbins will be on campus April 4-5 as the SECS computer science honoree for the DU Master Scholar Program this year.

Brian Robbins is the Founder of Riptide Games, an iPhone and next-gen mobile game development studio based in Denver, CO. He has spent most of his career pushing the limits of emerging tech markets and is frequently evaluating new technology and markets for unique ways to entertain people. Brian has programmed over 100 games and still finds time to dig into coding whenever he gets a chance.

In addition to running Riptide Games, Brian is a frequent speaker and industry conferences, and is heavily involved in volunteering within the game industry, primarily through his work with the International Game Developer’s Association (IGDA). He currently serves as Chair of the IGDA’s Board of Directors and as a Trustee for the IGDA Foundation. Prior to that he has chaired multiple Special Interest Groups and local chapters. He was awarded the IGDA’s MVP award in 2006 for his contributions.

Brian has a BS in Computer Science from the University of Denver, and an MBA from DU’s Daniels College of Business. He lives in Aurora, CO with his wife and their beautiful daughter.

The AI Model

- Execution Management
- Strategy
- Decision Making
- Movement
- Animation
- Physics

GROUP AI

CHARACTER AI

World Interface
Thought problem

• Suppose we are writing an AI for a FPS
  • What might we take into account?

One way to approach the problem...

• if (health < 30)
  • Go find health
• if (health < 50) && (health nearby) && (no enemies)
  • Go find health
• if (better weapon nearby)
  • Get better weapon
• .......

Why is this approach difficult?

• Difficult to tune parameters
• Difficult to manage behavior / insert new behavior
• Difficult to debug
• Difficult to reason about behavior over time

What is the intent of this approach?

• The goal is to classify regions of parameters where particular behavior should be used
• One way to do this is a decision tree
• A decision tree is a tree where:
  • Each state in the tree represents a classification decision (test)
  • Each leaf in the tree represents an action to take
Enemy nearby?

Health < 20?

Ammunition?

Yes

No

Yes

No

Find Health

Find Enemy

Find Weapons

Attack!

Decision Tree properties

- In a balanced tree with depth $d$, $b$ choices, and $n$ states
  - $n = b^d$ states
  - A decision requires $\log_b(n) = d$ evaluations - $O(d)$ time
- The entire tree requires $O(n)$ memory
- In an unbalanced tree
  - A decision could require $O(n)$ evaluations/time

Enhancing decision trees

- Tests can be randomized
  - eg if dice roll is > 4
- Branchings can be shared
- Automatic algorithms for balancing without changing the decision made by the tree
- Rich expressive power

OR function

- True

- False

- 2

- 1

- 2
AND function

Pros and Cons of decision trees?

Higher level view: State

State as actions

### AND function

- **A**
  - False
  - True
- **B**
  - False
  - True
- **2**
- **1**

### Pros and Cons of decision trees?

- Enemy nearby?
  - No
  - Yes
  - Health > 20?
    - Yes
    - No
    - Ammunition?
      - No
      - Yes
      - Find Weapons
      - Find Enemy
      - Find Health
      - Attack!

### Higher level view: State

- What is the state of the game/character?
  - Everything that has to go into a save file...
  - ...but may include other implicit knowledge

- Class data members:
  - health, location, etc

### State as actions

- State can also be:
  - What task is being performed?
  - What stage of the task am I in?
  - Essentially, the steps of an algorithm
Draw a dragon

Trogdor game

How do the dragons move?

Finite State Machine

- Represents the states a system can be in, and the transitions that occur
- Requires:
  - A list of possible inputs
  - A list of possible states
    - An input state
    - A final state (optional)
  - A list of possible transitions
Only a FSM?

- A FSM can model a computer (Turing machine)
  - We could do all AI in FSM’s
  - (But, may not be easy)

Sample Problem

- Write out a FSM that implements the “right-hand rule”
- To escape a maze:
  - Put your right hand on the wall
  - Keep your hand on the wall and walk through the maze
- Will this escape ANY maze?

Answer

- States:
- Input:
- Transition:
Algorithm

• Repeat:
  • Walk forward
  • If you hit a wall:
    • Turn left
  • Otherwise:
    • Turn right

FSM (definition)

• States:
  • RHR (only one state needed)
• Input:
  • w (facing wall)
  • f (facing free space)
• Transition:
  • Turn left (on w input)
  • Step forward and turn right (on f input)

Air Hockey - what does this code do?

```c
if (pucky < 0.00)
{
  if (puckdx == 0)
  {
    if (fequal(curry, -1) && fless(fabs(currx), 0.1f))
    {
      if (currx < 0)
        newx = -0.15;
      else if (currx > 0)
        newx = 0.15;
      else if (0 == random()%2)
        newx = 0.15;
      else
        newx = -0.15;
      newy = -1;
    }
    else
    {
      newx = puckx;
      newy = pucky;
    }
  }
  else
  {
    newx = puckx;
    newy = pucky;
  }
}
else {
...
```

Air Hockey - what does this code do?

```c
if (currentState == kSetupLeftShot)
{
  if (fequal(currx, 0.15))
  {
    currentState = kInitialShot;
  } else {
    SetAIMove(0.15, -1);
  }
} else {
  if (currentState == kSetupRightShot)
  {
    if (fequal(currx, -0.15))
    {
      currentState = kInitialShot;
    } else {
      SetAIMove(-0.15, -1);
    }
  }
}
```
Air Hockey - what does this code do?

```c
if (currentState == kInitialShot)
{
    float puckx, pucky;
    float puckdx, puckdy;
    GetPuckLocation(puckx, pucky);
    GetPuckSpeed(puckdx, puckdy);
    SetAIMove(puckx, pucky);
    if (puckdy > 0)
        currentState = kPerfectDefense;
}
```

FSM Variations

- Hierarchical
  - eg Trogdor dragons
- Non-deterministic
  - eg Air Hockey AI randomly chooses which direction to fire

Computational Costs

- Full FSM is stored (1 copy for all characters)
- Storing current state is O(1) (for each character)
- A decision requires O(m) evaluations, where m is the number of possible branches

FSM discussion

- What type of behaviors are easy to model with FSMs?
FSM discussion

- What type of behaviors are hard to model with FSMs?

Reverse AI suggestion (Paul Tozour)

- As you play a game:
  - Observe the actions you take
  - Look at the context
- What would it take to build an AI for your own play?
- (Game design)
  - Are parts of the game not needed to succeed?
  - Why are they there