

# CS 134

Video Game “AI”

# Today in Video Games



**Ellie Gibson**

Contributor

@elliegibson

**EUROGAMER**

GET OUR AWARD-WINNING  
COVERAGE THE MOMENT  
IT'S POSTED

 Follow

I learned an astounding fact the other day. No, not the thing about how the literal translation of James Van Der Beek is "James Of The Creek." Although I did have to have a sit down for a couple of days after that. [1][2]

This fact was to do with videos of people playing games on YouTube, which are of course hugely popular - some of them, inexplicably, even more widely viewed than [that clip](#) of an Olympic sprinter racing a giraffe. [3] In fact, 48 per cent of YouTube gamers spend more time watching videos than playing games. [4] (Although, since that survey was conducted, many of the other 52 per cent have said they didn't really understand the full implications of their response, and in light of revelations about the potential economic fallout and the shambolic nature of the negotiation process, they would answer differently if asked again tomorrow.)

But what really made my jaw hit the floor, almost to the point where I could be mistaken for James Van Der Beek, was learning that 40 per cent of people who watch games on YouTube have never played a video game. [5]

# Homework Questions

Any questions about the homework?

# Goal of Game AI

## Bioshock Infinite



## Pacman

<http://gameinternals.com/post/2072558330/understanding-pac-man-ghost-behavior>

# Goal of Game AI

- Is Elizabeth really revolutionary?
  - No! Very simple tricks were done.
- AI characters should act in predictable, human ways
- AI characters should make the same types of mistakes as a human

# Goal of Game AI

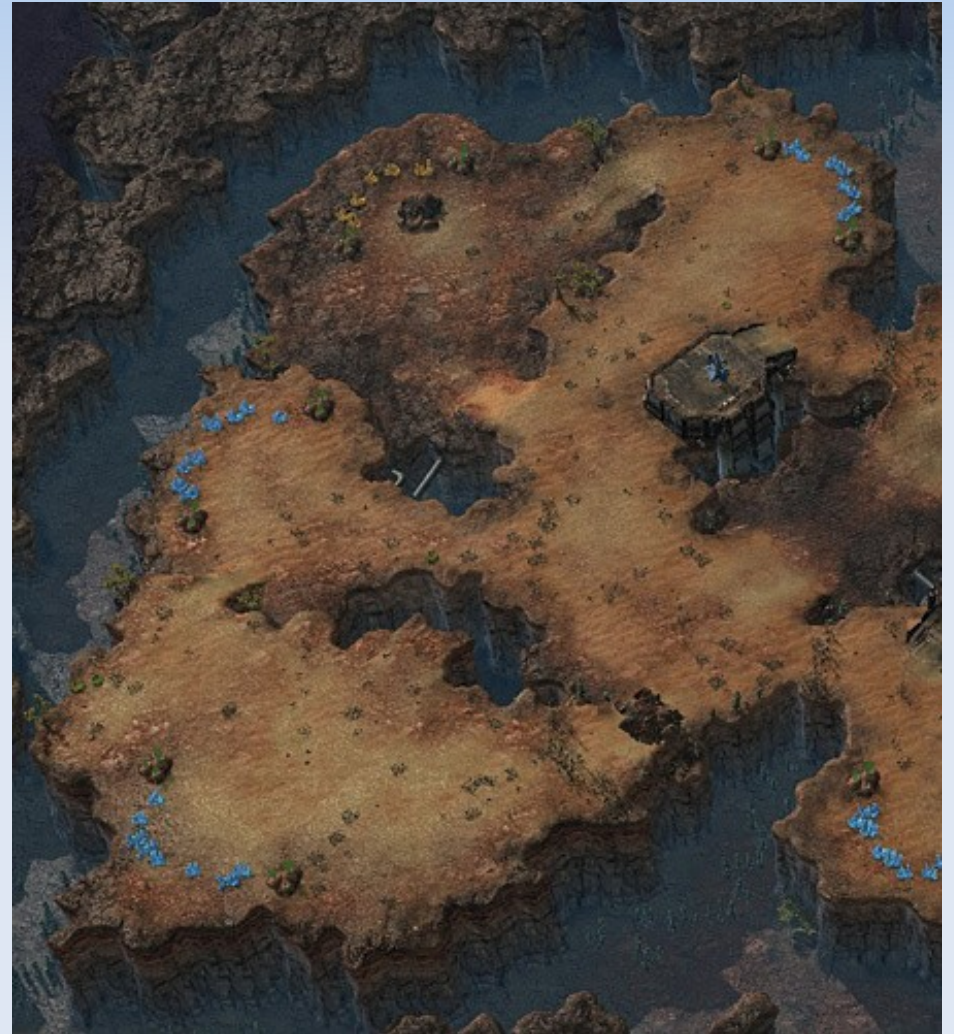
- Goal is not to create AI as CS thinks about it.
- Goal is create AI as designers think about it!
  
- Programming is a last resort
- Things to jump to later on:
  - Genetic Algorithms
  - Neural Networks
  - Decision Trees

# Smart World

- Think about the behaviors you want
  - What is data?
  - What is defs?
- Put everything you can into defs in the world
  - Where AI can walk
  - Where AI should defend
  - Special animations to play
  - etc.

# Smart World

- Good base building locations
- Best enemies to attack a location with
- etc.
  
- Yes, this data is all getting hardcoded. But that means it is easy to modify!





# Smart World

- What information was put in for Elizabeth?
- Interaction points
- Events
- Specific things that can get unlocked



# Smart World

You can get a lot of mileage out of putting intelligence directly into the world

Biggest limitation is that someone has to manually put all the behaviors into the game

# Game Loop

- AI has two distinct parts:
  - Decision making
  - Decision execution
- Decision making does not need to happen all the time
- Decision execution must happen every frame

# Game Loop

```
Player player;
Camera camera;
Enemy[] enemies;

// The game loop (inside main)
while (!shouldExit) {
    // Message pump, deltaTime, etc.

    player.Update(deltaTime);
    camera.Update(deltaTime);
    for (Enemy enemy : enemies) {
        enemy.Update(deltaTime);
    }

    // Drawing
}
```

- The current game loop is calling “enemyUpdate” on every single enemy.
- Inside enemy update you will put AI logic.

# Game Loop

```
// inside class Enemy

public void Update(float dt) {
    anim.Update(dt);

    // Simple logic, move toward player
    float deltaX;
    float deltaY;
    if( x < player.x ) {
        deltaX = ESPEED;
    } else {
        deltaX = -ESPEED;
    }
    if( y < player.y ) {
        deltaY = ESPEED;
    } else {
        deltaY = -ESPEED;
    }

    // Actually move!
    x += deltaX * dt;
    y += deltaY * dt;
}
```

- The current game loop is calling “Update” on every single enemy.
- Inside enemy update you will put AI logic.

# Game Loop

```
// inside class Enemy

public void Update(float dt) {
    anim.Update(dt);

    // Simple logic, move toward player
    float deltaX;
    float deltaY;
    if( x < player.x ) {
        deltaX = ESPEED;
    } else {
        deltaX = -ESPEED;
    }
    if( y < player.y ) {
        deltaY = ESPEED;
    } else {
        deltaY = -ESPEED;
    }

    // Actually move!
    x += deltaX * dt;
    y += deltaY * dt;
}
```

- Replace the movement logic with smarter AI.

# Game Loop

- In your update function add logic like this:

```
if (needANewDecision) {  
    makeNewDecision();  
}  
executeDecision();
```

# Game Loop

- Why have this distinction?
- Planning a decision is often slow.
  - Could take multiple frames!
- Executing the plan is quick and easy.



# Game Loop

```
// inside class Enemy

public void Update(float dt) {
    anim.Update(dt);

    // Move to target, choose new target
    // when needed
    if (x == targetX && y == targetY) {
        requestNewTarget();
    }
    float deltaX;
    float deltaY;
    if( x < targetX ) {
        deltaX = min(ESPEED, targetX - e->x);
    } else {
        deltaX = -min(ESPEED, e->x - targetX);
    }
    if( e->y < targetY ) {
        deltaY = min(ESPEED, targetY - e->y);
    } else {
        deltaY = -min(ESPEED, e->y - targetY);
    }

    // Actually move!
    x += deltaX * dt;
    y += deltaY * dt;
}
```

- Replace the movement logic with smarter AI.
- Note: if you are using pathfinding, you will need to have a full path instead of just a single target position.

# Game Loop

- Any requests need to be handled outside the tick in your game loop.
- Think of this handling like another message pump!
- Handle as much as you have time for.
  - Budget time based on desired framerate.

# Game Loop

```
// Globals!
Queue<AIRequest> aiRequests;

// In gameloop
long curTimeMs = System.nanoTime();
while (!aiRequests.isEmpty()
    && System.nanoTime() - curTimeNs
    < 5000000) {
    AIRequest last = aiRequests.remove();
    handleAIRequest( last );
}
```

- Handle only as much AI requests as time is available.
- Potentially an AI request may not get handled for many frames!

# Game Loop

Questions?

# Decision Making

- Remember, you just need to create a fun challenge
- Predictable, but not too predictable is a common goal
- Semi-randomness



# Decision Making

- Semi-random behavior can cheaply add variety.
- Instead of just always making a single decision, you can randomly choose from a weighted list.

# Decision Making

- Aggressive Thug
  - 50% Charge
  - 35% Combo
  - 15% Retreat
- Opportunistic Thug
  - 15% Charge
  - 35% Combo
  - 50% Retreat
- Timid Thug
  - 5% Charge
  - 5% Combo
  - 90% Retreat
- Ranged Thug
  - 40% Charge
  - 20% Combo
  - 40% Retreat

# Decision Making

- Call `Random.nextFloat()` to get a random number
  - For C++: `rand() / (float)RAND_MAX` will get you a float from 0 – 1.
    - `<.5` Charge
    - `<.85` Combo
    - `else` Retreat
- Store the decision until it is done.
- Inside `Enemy.Update()`, switch on the decision and do the right behavior.



# Decision Making

Questions?