# Game Engine Selection

Andrew Haydn Grant
Technical Director
MIT Game Lab
September 3, 2014

# Games Are Software

- UI
- Back end
- Design/Spec/Customer
- Features
- Bugs
- Task lists
- etc.

# Games Are Software++

- UI ***must be intuitive***
- User testing
- “Fun”
- User testing
- Gameplay Difficulty
- User testing
- Emotional Impact
- User Testing

# Preview

- Why Use A Game Engine?
- Criteria For Game Engine Selection
  - Dealbreakers
  - Nice to have
- Engines
  - The Good
  - The Nooses
- Final Word
  - How To Learn An Engine
  - All Software Sucks
  - Engine Assignment Mechanic

# Tinker Toys vs Vision

This is what I WANT.

How do I make it with these?

This is what I HAVE.

What can I make?

# Definition of “Hard”

“Actually, I don’t care how hard it is.
How long will it take?”

# NOT Writing Code

- Coding Is *Slow*
    - think
    - implement
    - debug
    - integrate
    - debug
    - debug
    - debug

# NOT Writing Code

- Coding Is *Slow*
  - think
  - implement
  - debug
  - integrate
  - debug

- So Write Less Code
  - Paper Prototyping
  - Iterative Design & Testing Early
  - Game Engines

# Why A Game Engine?

- Time
  - Avoid reinventing the wheel
  - Avoid certain kinds of bugs
  - Define general direction of your architecture
- Inspiration
  - Use the feature list as a set of possibilities
  - Use the feature list as a set of limitations

# Preview

- Why Use A Game Engine?
- Criteria For Game Engine Selection
  - Dealbreakers
  - Nice to have
- Engines
  - The Good
  - The Nooses
- Final Word
  - How To Learn An Engine
  - All Software Sucks
  - Engine Assignment Mechanic

# Game Engine Selection

- It’s an important decision
- But don’t stress about it too much.
  - No engine is perfect.

# Preview

- Why Use A Game Engine?
- Criteria For Game Engine Selection
  - Dealbreakers
  - Nice to have
- Engines
  - The Good
  - The Okay
  - The Nooses
- Final Word
  - How To Learn An Engine
  - All Software Sucks
  - Engine Assignment Mechanic

# Primary Selection Criteria
## (dealbreakers)

**Cost**

**Free**

**vs**

**Painful**

**vs**

**Impossible**

# Primary Selection Criteria (dealbreakers)

## Power: Can We Build It?

- Fundamentals only
    - Ignore bells & whistles
- 3d/2d
- Publishing platform
- Input methods
- Other known requirements

# Primary Selection Criteria
(dealbreakers)

## Ease of learning

- Search engine friendly
  - Support community!
- Tutorials & Documentation
  - Support community!
  - In-house experts?
    - A person knowing the engine is only useful if that person WELCOMES being a teacher.
- Learning Curve

# Primary Selection Criteria (dealbreakers)

## Ease of Use

- Strongly Typed Programming Language
  - Compile-time error detection
  - Free Communication Channels
    - Auto-complete code editor
    - Easier Integration

# Close Enough

## If you know one, you know the others:

- Java
- C#
- AS3
- Haxe

## Extra bonus:

- C++

If you know this, you know all of the above.

# Not Close Enough

**Java != Javascript**

# Primary Selection Criteria
(dealbreakers)

## Ease of Use

- Source control friendly?
- Debugging?

# Primary Selection Criteria (dealbreakers)

## Robust Product

- Few bugs
  - Hard to analyze quickly
  - Cues:
    - Strong Community
    - Large Community

# Preview

- Why Use A Game Engine?
- Criteria For Game Engine Selection
  - Dealbreakers
  - Nice to have
- Engines
  - The Good
  - The Nooses
- Final Word
  - How To Learn An Engine
  - All Software Sucks
  - Engine Assignment Mechanic

# Secondary Selection Criteria
(nice to have)

## Ease of Use

- Asset pipeline
- Source Code Available?
- Code IDE?
- World Editor?
- Profiling?

# Secondary Selection Criteria
(meh)

## Power: Bells And Whistles

- Rendering Speed
- Pathfinding
- Physics
- Shaders
- Shadows
- Particle Systems

# Criteria I Don't Use

- Scripting Languages
- Beautiful Games

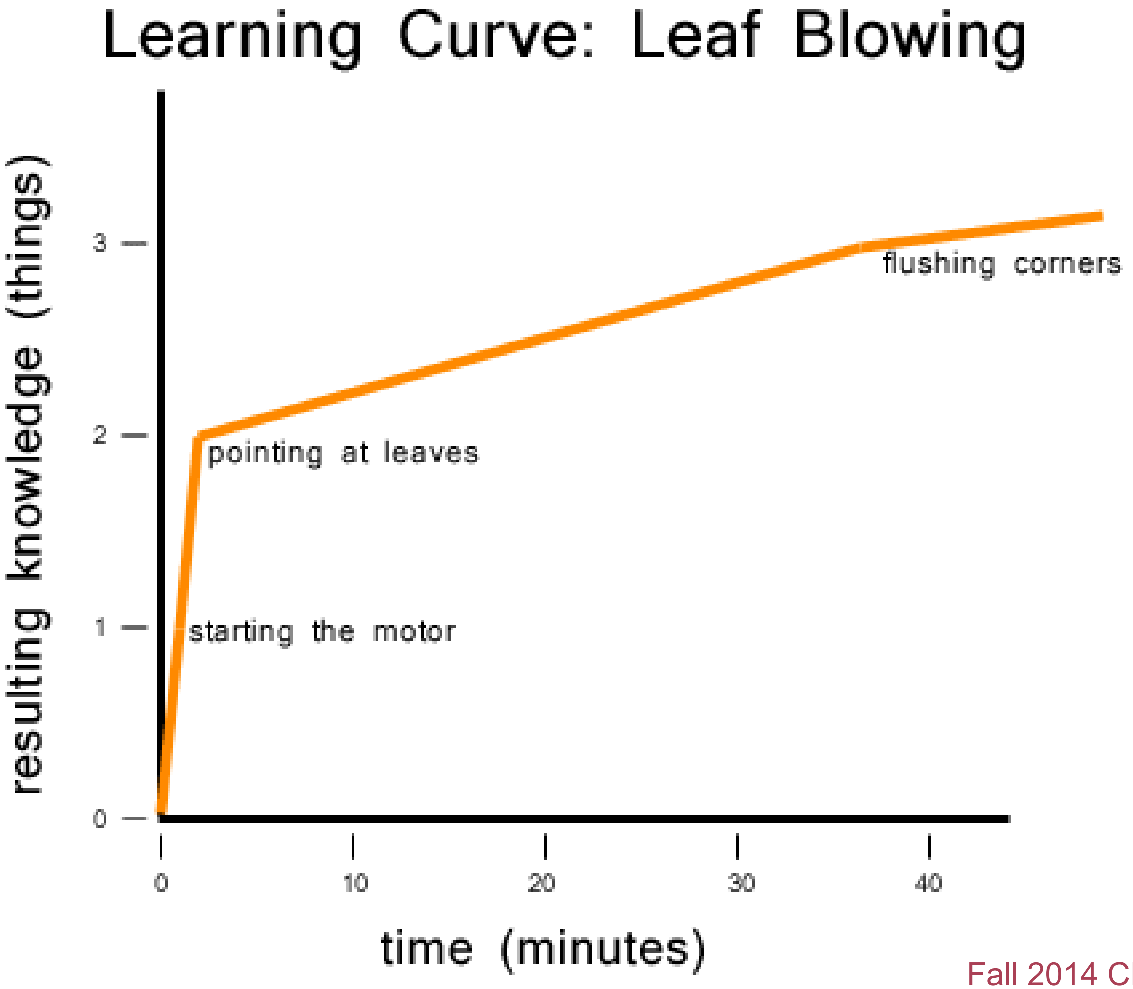
# Preview

- Why Use A Game Engine?
- Criteria For Game Engine Selection Dealbreakers
  - Nice to have
- Engines
  - The Good
  - The Nooses
- Final Word
  - How To Learn An Engine
  - All Software Sucks
  - Engine Assignment Mechanic

# Why Flixel?

- Free
- Publishes to web in Flash
- Robust
- IDE
- All source visible
- Strongly typed language
- Simple object oriented architecture
- Excellent for 2d sprite-based Action

# Learning Curve: Flixel



# Why Not Flixel?

- Not so good at heavy GUI work
- Falling usage
    - Adobe is insane

# Why Unity?

- Free
- Publishes to web
- Excellent Community
- Robust
- IDE
- Excellent Asset Pipeline
- Strongly typed language
  - and two weakly typed ones
- Harder to learn than Flixel, but easier than almost everything else
- Simple, but unusual component-based architecture

# Why Not Unity?

- 3D
- Source Control/Merge
- Not so good at heavy GUI work

# Preview

- Why Use A Game Engine?
- Criteria For Game Engine Selection
  - Dealbreakers
  - Nice to have
- Engines
  - The Good
  - The Nooses
- Final Word
  - How To Learn An Engine
  - All Software Sucks
  - Engine Assignment Mechanic

# Nooses

- Haxe Flixel
  - AS3, Flash pedigree
  - Flixel Pedigree
  - Untried
- Phaser
  - Education Arcade
  - Javascript
    - bleah- use Typescript!
  - Untried

# Learning A Game Engine

- Start with a tutorial
- Try something very small
- Skim the docs
- Try something harder

# All Software Sucks

But we still use it.

# Analysis Assignment

- You can trade game engines.

- Spend no more than 4 hours learning your game engine.
  - If you finish the tutorials, start making an *Asteroids* or *Space Invaders* clone.

- Bring your experiences to class Wednesday (Sept 10)
  - … and a development machine!

# Game Engine Selection

Andrew Haydn Grant
Technical Director
MIT Game Lab
September 3, 2014

CMS.611J / 6.073 Creating Video Games
Fall 2014