Foundations of Interactive Game Design (80K)

Lecture 9
Today

- Quiz
- Operational logics
- Reminder of what’s coming up
Does anyone recognize this game?

Adventure first real narrative game, we’ll talk more about it

Moving through complex spaces, gathering and using objects, solving puzzles
How do you make a game like that on a machine designed for Pong and Combat?

This is a platform issue, which we'll also talk about more. Working within your platform constraints is one of your main challenges as a designer.

You are standing at the end of a road before a small brick building. Around you is a forest. A small stream flows out of the building and down a gully.

> enter

You are inside a building, a well house for a large spring.
Adventure is the first graphical adventure game

Let's take a look
Adventure manual

This was the era when reading the manual could be assumed, though the instructions were generally as simple as those printed on arcade cabinets.
Demo
What did you notice about playing Adventure?
Playing *Adventure*

- Moving, running into walls
- Running into objects, picking them up
- Running into enemies with objects (combat)
- Running into objects with objects (keys)
- Bridging walls with special object
Does this sound familiar from somewhere?
Event-based programming
This looks simple

But it introduces much that is important to games
When fruit collides with a wall, this is an "event"

When a player clicks on fruit, this is an "event"
Event-based programming

- A simple idea: Wait for a specific event, then do something
- Learning to decompose problems this way will improve your Game Maker ability
- Games update everything as fast as they can (calculating physics, rendering reflections, updating AI, etc) and monitor for events based on ongoing action and player input
Back to our story
The Legend of Zelda, Shigeru Miyamoto, Takashi Tezuka, et al., 1986–7

Adventure games evolved, Zelda
Demo
What did you notice about playing *The Legend of Zelda* compared with *Adventure*?
Zelda versus Adventure

- More complex world
- More complex enemies and attacks
- More objects, inventory
- Resource management, vendors
- Running into things still fundamental: moving through world (boundaries), picking up rewards, fighting enemies
Let’s take a step back
At GDC 2011, Clint Hocking and others were asking this “big question” -- “How do games mean?”

Clint’s answer was, in part, to look at a particular game of Go that can be seen as a turning point away from Japan’s traditional past.

The young, winning player took an “ugly” placeholder move in order to buy himself two days to think. It could be seen as ruining the beauty of the game, but it could also be seen as the key to his win. This match is written about in the book The Master of Go.

But let’s start with a simpler question.
What does this mean?
Let’s consider a simpler example. Okay, pretty clearly this means “one thing runs into another”
And *how* does this mean?
It's not the same way that a painting would represent the same thing,
or this

or the way a movie would represent it,

or even the way that the videos embedded in this slide show present it.
Why is it that digital media can create worlds that feel present, and alive, and use that as a foundation for meaning-making?

In a platformer we can have the experience of touching virtual objects over and over

not just because we have a presentation of the game state, which expresses meaning much like a painting or movie

but because we also have underlying computational processes that support it
How does collision detection mean?

• Visually: A ball runs into a paddle, then something happens (it bounces off). Like how meaning happens in a movie.

• Experientially: As we play, an experience unfolds. Balls keep running into paddles, from different angles, and keep bouncing off — we get a feel for how the world works.
The foundation of game meaning

- The *experience* of collision detection is only made possible by an underlying process
- There are many specific collision detection algorithms, but all support an abstract process wedded to a communicative goal: when two virtual objects touch, something happens
These are operational logics
Operational logics

- A communicative goal — “virtual objects can touch”
- combined with an abstract process — “when two coordinate spaces overlap, do something”
- supporting ongoing media (re)presentation and audience experience
Implemented in many ways

• When you play *Pong* on an Atari VCS, the 2D collision detection is implemented *in hardware*.

• When you build a game using XNA, the 3D collision detection is implemented *in software*.

• Obviously, implementations differ fundamentally, but the logic — that virtual objects can “touch” — is the same.
When fruit collides with a wall, this is an example of an operational logic.
Further meanings

• Basic operational logics are articulated with game systems and themes
• Meaning approaches become conventional: running into walls, picking up objects
• Games build on this conventional knowledge like film builds on our familiarity with cross-cutting, etc
What are creative ways to tap into this experience of meaning?

How else might we use this way of making meaning?
Passage demo
What did you notice about playing Passage compared with Adventure and Zelda?
Passage

- Collision detection — and other elements of spatial navigation — re-mapped to options and constraints in life choices
- Splits apart conventional, almost-invisible logic — same abstract process, new communicative goal
- Could easily be made in Game Maker...

Passage changes the meaning of collision detection, or overloads it