Mid-Semester Feedback

• We’re happy that you’re happy – lots of 4 and 5s!
• Workload is about expected
• Top complaint: “I feel behind”
  – Retry system has you covered
  – …although we did warn you about the snowball
• Suggestions to fix this are welcome!
Special Topics Lectures

• Next week we’ll be covering:
  – Advanced graphics (with a brief on OGL integration)
  – Networking

• The rest of the schedule will be posted online
M2 Initial Feedback

- Physics is fun!
- M3 will be a full-fledged game!
- Make sure to come to hours if you’re spending huge amounts of time debugging
  - This weeks topics can be a bit tricky
M2 Initial Feedback

• Help! My objects are falling through the floor!
  – The impulse applied when restitution=0 should exactly cancel velocity along axis of collision
M2 Initial Feedback

- Help! Objects are sinking into each other when I stack them!
  - Run collision resolution multiple times per physics step until stabilization
  - Make sure to cap the number of checks per tick
Final Design

• Monday November 9th - 15 or 30 minute meeting with the TA staff to talk about your final
  – 30 if you’re in a group of 2 or more

• You’ll be telling us:
  – Why your game is worth making
  – What engine features you’ll implement (per person)
  – Whose engine you’ll build off of
  – If 3 or more, how you will use version control
CS1972: 3D Game Engines

• Running next semester!
  – Timeslot probably 3p-5:20p Wed
• Two pre-requisites:
  – Software engineering: 1971, 32, or 33
  – Graphics: 123 or a winter assignment (maybe)
• Topics include physics, world/level representation, pathfinding over navigation meshes
• 4 projects: warmup, minecraft, platformer, final
• http://cs.brown.edu/courses/csci1972/
  – See the website from last semester for more details
• You can run the project demos in /course/cs1972/demo
QUESTIONS?
Lecture 8

Content Management II
GAME LOGIC MANAGEMENT
Recap

• Why are we doing this again?
  – Make changes to assets without recompiling
  – Prevent executable bloat

• Some content is already separated
  – Graphics are out
  – Maps are out
What is Game Logic?

• Game logic dictates the rules of your game
• Consider a chess game
  – Maps set up a chess problem
  – Images define how the game looks
  – ...but game logic controls how the pieces move
• You probably hard code this

```java
cc = new CollisionConstraint();
rc = new RaycastGroup();

player.setx(320);
player.sety(100);
Hitbox boxA = new RectHitbox(player, 32, 64);
me = boxA;
player.setMass(1);
player.addChild(boxA);
cc.addChild(boxA);
gameLayer.addChild(player);

GameObject ground = new GameObject();
ground.setx(-200);
ground.sety(480 - 32);
Hitbox boxC = new RectHitbox(ground, 1000, 640);
ground.addChild(boxC);
ground.setMass(0);
cc.addChild(boxC);
gameLayer.addChild(ground);

derp = new GameObject();
```
What is Game Logic?

• Not quite the same as game data...
  – Game data could describe item names, character stats, etc, without defining behaviors
  – But it’s a blurred line and many solutions cover both
• Could even be the parent of many different assets
  – Cutscenes
  – Lightweight menus
  – Etc…
Why not hard code logic?

• Programmers aren’t game designers
  – And they shouldn’t be game designers’ personal assistants either

• Game logic should be reusable
  – Many common behaviors can be factored out
  – ...even across multiple games

• Convenience — it’s faster
Content Management II

SOME POSSIBLE SOLUTIONS
Put Logic in Config Files

- Logic is defined in external files, sort of like Tac maps
- Good for static information
  - Unit stats, like how much health tanks start with
  - Game settings, like screen size or default level
- Trickier when trying to define interactions
  - How would you define the M2 grenade?
Embed a Scripting Language

• Logic is defined by a scripting language, then loaded at runtime
• Good for complicated behavior
  – Wouldn’t Tac AI have been so much easier?
• Lots of overhead
  – Need a well-designed API
  – Designers have to learn API
Make a Game-specific Editor

- Logic is defined in a settings application tailor-made for the game
- Very easy to use
  - Engines can even be aimed at non-programmers (RPGmaker etc)
- Inherently not portable
  - And you have to spend all that time writing it again!
  - What if we had a general editor for the engine instead?
What Features do we Need?

• Need to be genre-agnostic
  – Because our engine is too
  – …although feel free to pursue other approaches for final

• Need to define entities themselves

• Need to define entity interactions
  – For instance, how would we make a weighted button?
QUESTIONS?
Lecture 8
Entity I/O
WHAT’S ENTITY I/O?
Introducing Entity I/O

• Strategy for logically connecting entities
• Connections are defined by designers and dictate game logic
• Popularized in the Source engine

When button is depressed, release rock
Inputs and Outputs

• Entities have outputs
  – `button.onPress()`
  – `timer.onFinish()`
  – `camera.onSeen()`

• Entities have inputs
  – `door.doOpen()`
  – `bomb.doExplode()`
  – `lights.doToggle()`
Some Example Uses

- **Light switch**
  ```
  switch.onFlip() → light.doToggle()
  ```

- **Puzzles**
  ```
  ocarina.onPlay() → timeCube.doRaise()
  ```

- **Automatic door**
  ```
  camera.onPlayerSeen() → door.doOpen()
  ```
Connecting Entities

• An output can be connected to an input to define a behavior
• An output is connected to any number of inputs
  — This might seem backwards, but think of connecting wires
• Programmer defines inputs/outputs
• Designers connect inputs/outputs

onPush() -> doOpen()
Case study: Gunpoint

- Has several devices as entities
  - Light switches, alarm systems
- Allows you to dynamically change the connection between each one
  - Light switches can be modified to open up locked doors
- [https://youtu.be/T6a0WR2-tLg?t=120](https://youtu.be/T6a0WR2-tLg?t=120)
Invisible Logic Entities?

- Entities need not be physical
- What about an invisible hitbox that performs some logic when the player enters the box?
- …and a Timer doesn’t physically interact at all!
Special Logic Entity: Sensor

- Sends a signal when some other entity touches it
- Not necessarily physical
  - Triggers when a player enters an area, etc
- Can you think of some uses?
Special Logic Entity: Relay

- Think of it as a pipe with a valve
- Relays a signal from one entity to another
- ...but only when it’s received some other signal
Other Special Logic Entities?

• There are tons of things that games share in common

• Can you abstract out these “standard entities” into your engine?
  – You’ll have to do at least 2 standard entities for M3

• Could you give someone JUST your engine and the level editor and have them make a game?
Engine Objects

- An Output, to send a pulse to all attached inputs
- An Input, to hold and run arbitrary game code
- Potentially a Connection to go between them
  - Allows passing connection properties as input arguments
Setting Up Connections

• Connections are defined outside the game code
  – These are loaded from data
• Data needs to be turned into your Output and Input objects
• Then these objects fire at their appropriate times in game
Some Sample Contracts

```java
public class Output {
    private List<Connection> connections;
    public void connect(Connection c) { connections.add(c) }
    public void run() { for (Connection c : connections) c.run() }
}

public class Connection {
    private Input target;
    private Map<String, String> args;
    public void run() { target.run(args); }
}

abstract public class Input {
    public abstract void run(Map<String, String> args);
}
```
• Need to read connections from data files and then initialize them
• Entity subclasses can have outputs/inputs
  - But there might be some common ones: onCollide etc
• Inputs need game-specific logic

```java
public class Door extends Entity {
    private Output onDoorOpen;
    public OpenDoorInput doOpenDoor;
    private void openDoor() {
        // … door open code …
        onDoorOpen.run();
    }

    class OpenDoorInput extends Input {
        public void run() {
            openDoor();
        }
    }
}
```
QUESTIONS?
Lecture 8
Tips for M3
Constructing a world

• We have new support code!
  – CS1971LevelReader
  – LevelData

• Use the properties of LevelData to populate your world
Constructing a world

• Keep a reference of the available classes/Entity types you have
  • `Map<String, Class<?>>`

• Keep a reference of the entities in your level
  – `Map<String, Entity>`
Constructing a world

- Iterate over all the entities in your level object and translate that into actual `Entity` objects
  - Use your `Map<String, Class<?>>` to create them
  - It’s your engine – make them all have a special constructor or initializer!
- Iterate over all the connections in your level
  - From your `Map<String, Entity>`, connect your `Entity` objects together
Reflection and Friends

- Avoid `Class.forName()`
  - What happens when code is re-factored?
  - Have to sync data and code
  - You may be tempted to use this this week – don’t!

- (Also, cs1971 publisher obfuscates demos, breaking most reflection)
Connections, not Logic Gates

- Connections send discrete events, not electrical signals
- These events occur at some exact point in time; they don’t become true
- Can’t be “evaluated” like state machine transitions
  - So they require a different structure
Tips for M3

JAVA TIP OF THE WEEK
Breaking is Awkward

- Let’s say we have nested loops
- A `break` will only escape the innermost loop
- So we normally need some dumb boolean to keep track

```java
// find the first occurrence of 0
int row, col;
boolean found = false;
for (row=0; row<rows; row++) {
    for (col=0; col<cols; col++) {
        if (data[row][col] == 0) {
            found = true;
            break;
        }
    }
    if (found) {
        break;
    }
}
```
Introducing Labeled Breaks

• Code blocks can be labeled
• A **break** can be made to escape to a certain labeled block
• Can also use this strategy with a **continue**

```c
// find the first occurrence of 0
int row, col;
search:
for (row=0; row<rows; row++) {
  for (col=0; col<cols; col++) {
    if (data[row][col] == 0) {
      break search;
    }
  }
}
```
Other “Fun” Stuff

- Arbitrary blocks of code can be labeled
- Therefore you can have an arbitrary break
- Whee! It’s like a goto!
  - But don’t use it like one
  - Can only jump within the encapsulating block

```c
myLittleGoto: {
    // whatever code blah blah
    if (check) {
        break myLittleGoto;
    }
    // do some other stuff
    return;
}

// execution ends up here if // check is true!

// never do this ever!!
```
GAME DESIGN 7

Story
Advantages of Story

• Provides a clear beginning, middle, and end to the action
• Provides motivation for the player
• Players can take the identity of a character
• Story can create a sense of immersion
Disadvantages of Story

• Story writing takes time and care, similar to artwork
  – Plots can become convoluted
  – Plots can be bad
• Story requires heavy investment in visuals and audio assets
• Storytelling can slow gameplay
• Replay value
  – Why play the same game again?
  – Why read the same book twice?
The real story is the player’s

- Many believe a designer’s goal is to write a compelling story
  - We know this is wrong!
  - A mediocre story can sell if the player becomes immersed in the experience

- Good characters and story can help with immersion
Remember...

- We create the world of the game. We bring the player into that experience. And they fill it with their emotions.
LET’S TALK STORIES!
A Rule of Thumb

• “There is no original story” — How to Read Literature Like a Professor
  — Many fantasy RPGs drew from their pencil-paper ancestors (many of which drew from J.R.R. Tolkein’s The Lord of the Rings)

• Many stories draw fundamentally from religious roots as well
The Three Act Story

• All stories must have:
  – The Beginning
  – The Middle
  – The End

• This is a massive oversimplification, but it’s a good way to approach stories in games
The Beginning

• Many writers start by creating a lush world
  – This does not work in games

• The story and game starts the moment a problem is presented to our hero
Some beginnings:

- *Limbo* — You wake up in a forest...
- *Skyrim* — A dragon is giving you a second lease on life, don’t waste it!
- *Halo* — Wake up, get to the bridge
- *BioShock Infinite* — Find Elizabeth
- *Slender: The Eight Pages* — Find 8 pages
What to include?

• The best beginnings include:
  – Very immediate threats and obstacles that will relate to the more overarching threat or obstacle that will define the story
  – Tools to overcome these immediate threats and obstacles

• These lend themselves well to an environment where your player can learn how to play your game as well
The Middle

• Forms the bulk of the story
• This is where you introduce the more finely grained details of the world
  – If you wouldn’t mention it when describing the story in one sentence, it probably goes here.
  – Supporting characters
  – Specific locations within the scope of your universe
• Relate the hero’s action to the overarching background
The Middle
The End

• The conflict reaches some sort of resolution
  – Does not mean the conflict is fixed/solved!
• The player should feel *something* and remember it
  – Achievement
    • Civilization
    • Star Wars: Battlefront
  – Victory
    • Portal
    • Street Fighter
  – Shock
    • Freedom Bridge
  – Loss/Sadness
    • Halo: Reach
  – This is by no means an exhaustive list
Who is our hero?

• The more a player projects themselves into the protagonist, the better
  – This does not mean the protagonist has to be *like* the player
• Does not have to be the entity that the player controls
  – Ex: Starcraft II’s Jim Raynor, Sarah Kerrigan, and Zeratul
Character Growth

• Characters are the first part of your game that the player will grasp onto emotionally, so they need to be dynamic or the players will detach

• In literature, characters grow through some sort of internal/emotional change

• In games, characters grow by power-ups and level-ups.
  – This is not the same thing
  – The take: It’s harder to develop characters in games.

• This leads to the problem of interactivity
Conflict

• Classical classifications:
  – Man against man
  – Man against nature
  – Man against self

• Other classifications
  – Man against machine
  – Man against fate
  – Man against supernatural
  – Man against god
Antagonist...generally speaking

- Stories have some sort of “enemy”
- Does not need to be a single individual
- Basically whatever creates the conflict or whatever obstacle exists
Know your audience!

• What will your game be rated?
• Based on the rating, what can you incorporate into your game?
• Example: Movie and TV heavily regulate the use of profanity
  – How many bad words can you squeeze in to maintain a PG-13 rating?
Cultural Gaps

• Visual novels are very popular in Japan
  – Some are similar to dating simulations
  – No market in the United States
• Germany censors extreme violence
Story writing is hard

• If you don’t think you’re good at it, don’t worry
• Here is an example process:
  – What do you want your player to feel like?
    • Explorer? Conqueror? Soldier? Underdog?
  – What kind of universe does that game exist in?
    • What is the setting? Does it all take place in one town? One continent? One planet? One timeline?
    • What are the rules of your world? Is there something supernatural about it? What is the state of the world’s technology?
  – What kind of protagonist will thrive in this world?
    • What obstacles are they good at overcoming that makes the player want to project themselves onto the protagonist?
    • What obstacles are they bad at overcoming that makes them grow?
The 7 Basic Plots

1. Overcoming the monster (Shrek, Legend of Zelda)
   – Protagonist sets out to defeat some evil force that threatens them or their homeland

2. Rags to riches (Cinderella, Fable 3)
   – Poor protagonist experiences wealth, loses it all, and gains it back by growing as a person

3. The quest (Lord of the Rings, Borderlands)
   – Protagonist and companions set out for an important location or object

4. Voyage and return (Finding Nemo, Halo)
   – Protagonist goes to a strange land, overcomes challenges, and returns with only the experience

5. Comedy (Much Ado About Nothing, Saints Row 3)
   – Light and humorous, triumph over adverse circumstance and a happy ending

6. Tragedy (Macbeth, Death Note, Spec Ops: The Line)
   – Protagonist falls from grace and becomes a villain, whose death is a happy ending

7. Rebirth (Despicable Me, Red Dead Redemption)
   – An important event causes the protagonist to change their ways and become a better person
There is no original story

• Tropes and Clichés work!
  – People relate easily to things they have seen
    • Alluding to (but not using directly) other experiences that you know elicit a certain emotion is one of the best ways to get the player to feel a certain way
  – Use in moderation, don’t make it the core of your story
    • Works well as a setting
Case Study: AC4: Black Flag

- Assassin’s Creed 4 puts the player in the shoes of a pirate
- Arrrr!
  - Simple sailing mechanics used to travel
  - Pirate ships, cannons, pistols, cutlasses
  - Swinging from ropes to board ships and plunder them
  - Crew sings sea shanties while you travel
- Every player who approached the game already understood how to be a pirate
  - This gave the designers a solid base from which to develop the main character and the story surrounding him
Story writing is hard

• Take away:
  – Your player approaches your story from the aesthetic, the overall “feel”
  – You approach your story from the details, the rules that govern your universe
  – You have to start with the overall effect you want to have on the player, and work backwards
Things to See

• *The Ultimate Guide to Video Game Writing and Design*
  – Flint Dille and John Zuur Platten

• *Game Design* (2004)
  – Bob Bates
  – Advice on genre specific design
QUESTIONS?
M2 Playtesting

Let’s do it!