Support Code Updates

• We updated the support code last week!
• Please redownload it!!
Next Week

• No class next Monday!
• No project due Sunday either
• Wiz I design checks this week
• Wiz I & Alc II standard retries due 10/14
Alchemy 2 Feedback

• Email your grader when you want your retries graded!
  – We won’t look at any retry handins until you email us

• Even if you’re not done, hand in something!
  – Knowing exactly what you’re missing makes getting the standard retry correct a lot easier
Alchemy 2 Feedback

- Resources (image files, .mp3s) should be at the same level as src and bin
- alc/
  - src/
  - bin/
  - resource/
    - wood.png
Alchemy 2 Feedback

• Files
  – Don’t use absolute paths
  – “/gpfs/main/home/<login>/course/cs1971/tac/resources/spritesheet.png” is bad
  – “resources/spritesheet.png” is good
  – Absolute filepaths won’t work when we/your classmates try to test your game
Alchemy 2 Feedback

• In your handin, you MUST include:
  – a copy of the rubric
  – how to verify each requirement
  – how many hours it takes to complete
• Next week, you will get playtesting points taken off for this
David’s Hours

• They were Sunday, 9pm to 11pm
• They are now Sunday, 10:00pm to 12:00am
QUESTIONS?
This week: Wiz 1!

• Real gameplay features!
• Level Loading or Level Generation!
• Actually fun!
Lecture 3

Graphics III

Hang in there!
ANIMATING SPRITES
Why Sprite Sheets?

• In Alc, we could have a separate sprite sheet for each element
• What about for animation?
• Index different sprites for different movements- but from the same sheet
AnimationBehavior

• Like the SpriteBehavior, but also responds to TickMessages
• Can inherit from SpriteBehavior
• Animate by drawing frames like a flipbook: one after another, then reset
  – Store a list of positions for frames
  – Calculate which frame to use based on elapsed time
• Update frame state on tick, draw correct frame on draw
  – Draw is read-only
• Use modulo (%) operator to start over
QUESTIONS?
Physics I

COLLISION RESPONSE
Collision response

• You know how to detect whether 2 objects collide
• How do we make objects respond in a physically believable way?
• General strategy:
  – Move objects out of collision
  – (also other stuff later)
Physics I

MINIMUM TRANSLATION VECTOR
Moving out of collision

- Many ways to move the ball out of the wall
- We want the minimum
- “Minimum translation vector” (MTV)
MTV in one dimension

• In 1D, convex shapes are line segments (intervals)
• These have a 1D MTV
  – Similar to overlap
  – But it has a sign
  – Different for each shape
• To correct their positions, move by half the MTV
Computing circles’ MTV

• Circle vs Circle
  – Sum of radii – $\text{dist}(\text{center 1, center 2})$
  – MTV is parallel to line connecting centers
Computing circles’ MTV (ctd)

• Circle vs Box
  – If Box contains circle center
    • Find $p =$ closest point on AAB edge from circle center
    • Length of MTV is $\text{Radius} + \text{dist}(\text{center, } p)$;
    • MTV is parallel to the X or Y axis
  – Otherwise
    • Clamp circle center to Box
    • Length of MTV is $\text{Radius} - \text{dist}(\text{center, clampedPoint})$
    • MTV is parallel to line connecting
Computing boxes’ MTV

• Only four possibilities
  – Move up (this.maxY − that.minY)
  – Move down (this.minY − that.maxY)
  – Move left (this.maxX − that.minX)
  – Move right (this.minX − that.maxX)

• Return the shortest
Static Objects

• Some objects shouldn’t move, even when collided
• When a static object collides, it doesn’t move
• When an object collides with a static object, it moves by the full MTV
Collision callbacks

- Pass in other GameObject

```csharp
class PhysicsBehavior {
    void onCollide(GameObject);
}
```
Collision callbacks

- Separate `Collision info object` (really a struct)
- Pass in the MTV
- Pass in which `Shape` collided
  - Enemies with weak points

```java
class PhysicsBehavior {
    void onCollide(Collision);
}

class Collision {
    final GameObject other;
    final Vec2f mtv;
    final Shape thisShape;
    final Shape otherShape;
}
```
How MTV’s Affect Our Engine

- Collision methods should return MTVs, not booleans
- Be careful with signs and argument order
  - Especially when reversing args for double dispatch
- Be careful when calculating the MTV between concentric circles, you might end up dividing by 0
Physics I

QUESTIONS?
Lecture 3
Map Generation

*It's possible that you might have a problem
Map Generation

MOTIVATION
Hand Crafted vs Procedural

Hand Crafted:
• Straightforward
• Predictable
• Time intensive
• Will be covered later

Procedural:
• Far more variety
• Can lead to janky edge cases
• Will be used this week
Procedural Generation

• Algorithmically generate your own maps
• Game side - experiment!
• Typically uses seeded random numbers
  – Ex. `Random r = new Random(seed);`
  – Calling `r.nextXXX();` some number of times will return the same sequence of numbers
  – The seed can be used to share or save the generated map
  – Used to generate seemingly-hand designed content
• Somewhat different than randomly generated
Constraint-based Generation

• Not just any random map will work
• Generated maps need to follow game-specific constraints
  – Dungeon crawlers require a path from entrance to exit
  – An RTS might require every area of the map accessible
  – What looks good, what’s fun, etc
• Constraints are baked into the algorithm
Simple Generation Algorithms

- Value noise/Perlin noise
- Space partitioning
- Exploring paths (random/drunken walk)
- Lots of resources online

- Make map generation as generic as possible
Map Generation

Space Partitioning
Space Partitioning

• Basic idea – keep splitting the map up into smaller subsections to create rooms
• Used to simulate the insides of structures
Space Partitioning

- Start with an empty rectangular grid.
Space Partitioning

- Pick a random index on which to divide the space along the x axis.
Space Partitioning
Space Partitioning

• Pick another index on which to divide, this time dividing along the other axis (in this case y).
• Use a different index for each split
Space Partitioning
Space Partitioning

- Keep dividing, switching between x and y until you hit some depth (3 here).
Space Partitioning

• Fill spaces with random sized boxes.
• Make sure boxes fill up more than half of the width and height of the space they occupy.
Space Partitioning

- Connect sister leaf nodes of the tree.
- If rooms don’t take up more than half their space’s width and height, you might get z-shaped hallways.
Space Partitioning

- Connect parent nodes.
Space Partitioning

• Keep on connecting up the tree.
Space Partitioning

• If the halls are too narrow, Increase width of hallways to create more open space.
Space Partitioning

- Now you have your series of connected rooms!
- But there’s more...
Space Partitioning

• Instead of always checking depth, have some branches of the tree stop early so you end up with more variation in room size.
Constraints

• Add a minimum width/height
• Prevents rooms from being too small and weirdly shaped
Space Partitioning

- Say you wanted to keep spawn and exit rooms far apart.
- During the first split, assign one side of the tree to Spawn and the other to Exit.
Space Partitioning

- At the bottom of the Spawn subtree, assign one room to spawn.
- Symmetrically for Exit subtree.
Space Partitioning

QUESTIONS?
Lecture 3

Greater Dog.

Level Loading
**Important Map Information**

- Size of map
- Locations of terrain (grass, desert, trees, etc.)
- Starting location of units, unit types, unit orientation (friendly or enemy)
- Location of sprites, on sprite sheet, for unique objects
File Parsing

• Good news: Mostly game-side
• Bad news: So many things can go wrong!
  – Map file can’t be opened
  – Map file is empty
  – Map file is a directory
  – Map file is a JPEG
  – Is a map file, but has inconsistent data
• We’ll show you how to handle this
Parse Safely

• Read in a line, then parse it, repeat
  – At least you can report the line count where an error happened
• Recommended classes:
  – BufferedReader (for reading lines)
  – Scanner+StringReader (for parsing each line)
• Catch exceptions
  – Throw your own LevelParseException
  – Report useful debugging information
• We require that your parser never crash!
QUESTIONS?
Lecture 3
Tips for Wiz 1
Tips for Wiz I

VIEWPORT – CENTER ON PLAYER
Bad Way

- Give the viewport a reference to the player
- On tick, update the viewport center to be the player’s transform
  - Too much game logic handled by the viewport
Better Way

- Give the player a `CenterBehavior`
- Give the `CenterBehavior` a reference to the viewport
- Each tick, the `CenterBehavior` sets the center of the viewport to its object’s transform
Tips for Wiz I

MINIMAP
Viewports as Minimaps

- Minimap can be just another viewport
- Viewports should be able to draw the world normally
- They can also be generic enough to draw the world as a minimap
  - `draw(GraphicsContext g);`
  - `drawMinimap(GraphicsContext g);`
- Could instead have a MinimapSystem that draws the bounding boxes of GameObjects
Tips for Wiz I

JAVA TIP OF THE WEEK
Speeding Up Your Code

• Component-based engines are more expensive
• Eclipse has a java profiler!
• jvmmonitor.org
• marketplace.eclipse.org/content/jvm-monitor
Asserts

• Asserts:
  – assert thisFunctionReturnsTrue(); // no error
  – assert thisFunctionReturnsFalse(); // error
  – Can be useful for debugging
  – Particularly MTVs
public void parse() throws LevelParseException {
    throw new LevelParseException("Uh-oh!");
}
Catching Exceptions

```java
try {
    map.parse()
} catch (LevelParseException e) {
    //...
} catch (IOException e2) {
    //...
} finally {
    //...
}
```
ALCHEMY 2 PLAYTESTING
Let's do it!