Announcements
Alchemy 1 is done!

• Initial feedback for Alchemy 1
  – Viewports are important – fix them now!
  – Panning/moving should be done on tick
  – Organize your projects well! If you’re unsure about your design, talk to a TA
  – Keep our support code in a separate folder. Helps us grade.

• Next week your game will really start to take form!
Don’t Forget Tic

• Retries should all be returned
  – Please send us an email after you hand in any retry

• A few more tips for the future...
  – Watch out for edge cases
  – Plan out game/engine separation before you start
Lecture 2

Hang in there!

Graphics II
WHAT'S A SPRITE?
THIS IS A SPRITE
Sprites as Bitmap Data

- “Raster” graphics
- Pre-constructed images dynamically placed on the screen
- Designed to represent one type of object in a game
  - Objects may reference different sprites depending on state
Sprites as Animation Data

-Sprites as a filmstrip
-Designed to represent frame-by-frame snapshots of a single game object
-Standardized padding, size, and spacing allows for easy drawing
Typical Sprite File Format

- Multiple sprites per file
- Additional information often (but not always) in config files:
  - Padding
  - Size
  - Locations of a particular object’s sprites
Formatting “Standards”
Keep In Mind

• Bounding box info and sprite info should be separate
• But keep in mind that they will need to coordinate with each other
IMPLEMENTING SPRITES
Sprite Loading

- You should only load a sprite sheet image once
  - Each behavior using the sprite maintains a reference to the sprite sheet
- Consider making a Resource class which loads in sprite sheets
  - Load in image
  - Handling image index for different sprites
  - Generalizable to other assets like maps, sounds, text, etc...
Drawing Sprites

• About `g.drawImage(...)`
• Rare exception to the no JavaFX rule:
  – You’re going to need to make a JavaFX image.
  – Pass the RELATIVE file path
• Your drawing routine should handle different padding and formats
Relative Paths

• For All Resource 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
Drawing Sprites

• Draw rectangular chunks from sprite sheet to the canvas
• Don’t cache sub images
  – It isn’t worth the space/time tradeoff
• Remember to draw from your single sprite sheet reference
SpriteBehavior

- Has a reference to sprite sheet resource
- Should implement `draw(GraphicsContext g)`
- Once it has a `GraphicsContext` object, it can draw itself
TransformBehavior

- Where should the SpriteBehavior draw the image?
- How big should the image be?
- TransformBehavior - stores a position and size
- The TransformBehavior is special
  - All game objects should have one, separate from their behavior list
QUESTIONS?
Lecture 2
Collision Detection

*It's possible that you might have a problem
MOTIVATION

Collision Detection I
Collisions have consequences

• Collision detection is central to the vast majority of games
• They’re very important
What do we want to collide?

- Points
- Circles
- Axis-Aligned Boxes
  - Covered soon™
- Convex polygons
  - Not covered
- Other shapes

\[ P_{o} \]
\[ C \]
\[ r \]
\[ \dim = (w, h) \]
DETECTION ALGORITHMS
Point-Circle

- Check if the distance between the point and the center is less than or equal to the radius
- \( ||P - C||^2 \leq r^2 \)
Circle-Circle

- Check if the distance between the two centers is less than or equal to the sum of the radii
  \[ ||C_1 - C_2||^2 \leq (r_1 + r_2)^2 \]
Point-AAB

- Check if the point is within range on each axis

\[ \min_x \leq p_x \leq \max_x \text{ AND } \min_y \leq p_y \leq \max_y \]
Circle-AAB

- Check if closest point to circle on AAB is in circle
  - Closest point: clamp C.x, C.y to \([\text{min.x}, \text{max.x}], [\text{min.y}, \text{max.y}]\)
  - Then just do point-circle collision
AAB-AAB

- Ensure overlap on each axis
- Project each box onto x and y axes
- Find all four *Interval*ls, test for overlaps
Projection

- Imagine a light source with parallel rays
- Shape is between light source and axis
- “Shadow” cast on axis is shape’s projection onto that axis
Creating Projections

• Find the axis you want to project onto
  – This should be a normalized vector (length 1)
  – Vec2d has a normalize method

• x axis = Vec2d(1, 0)

• y axis = Vec2d(0, 1)
Creating Projections

• To project a point, take its dot product with the projection axis
  – `double p = point.dot(axis)`
  – Store p for later

• `Vec2d` has a dot product method
Creating Projections

• To project an AAB onto the x axis:
  – Project the top left and bottom right points onto the x axis
  – Store the two doubles in an **Interval**
  – The space between them on the x axis is the projection (shadow) of the AAB on the x axis
Projections $\Rightarrow$ Collisions

• For each axis, check if the corresponding intervals overlap
  – There should be two intervals for each axis
• Intervals $A$ and $B$ overlap if and only if:
  – $A_{\text{min}} \leq B_{\text{max}}$ AND $B_{\text{min}} \leq A_{\text{max}}$
• If both axes overlap, the shapes are colliding
Interval Class

• Stores two projections

```java
public final class Interval {
    private double min;
    private double max;
    public bool overlap (Interval other)
}
```
Collision Detection I

COLLISION BEHAVIOR
Shapes

• AAB and Circle classes inherit from the same abstract class
  – Shape attributes
  – Implement collision checks
    • Point collisions are only for the mouse; no separate class needed
Collision Behavior

• Contains collision information for a game object
• Holds the specific Shape for that GameObject
Collision System

- Keeps track of all game objects that can collide
- Loops through all pairs of registered objects
- Checks if each pair is colliding
- If there is a collision, both are notified - only go through each pair once
Expanded Contract

public void tick(long nanosSinceLastTick);
public void draw(Graphics2D g);
public void collide(GameObject o);
Collision Debugger

- Easy way to test collisions
- Will give you stencil code
- You fill in the math
Lecture 2

Greater Dog

Tips for Alchemy 2
Removing Units

- Beware the `ConcurrentModificationException`!
- Consider a removal queue
  - This can be generalized to multiple phases of ticks

```java
/**
 * Cleans up the rectangles that are not visible
 */
protected void purge() {
    for (Rect r : rects) {
        if (r.p2.y > 480) {
            rects.remove(r);
        }
    }
}
```
Sprites

- You’ll need to have sprites in your game to make it pretty!
- Lots of sprites on the internet
- Stealing IP is fun and easy!
  - We do it every lecture
  - Be sure to call it fair use
Double Dispatch

• If you have a `Circle` and an `AAB` but only know that they’re `Shapes`, how do you determine which method to call?

```java
void testCollide() {
    Shape s = new Circle();
    Shape s2 = new AAB();
    s.collides(s2);
}

interface Shape {
    collides(Circle c);
    collides(AAB aab);
    collides(Shape o);
}

boolean collides(Shape o) {
    if (o instanceof Circle) {
        return collides((Circle) o);
    } else if (o instanceof AAB) {
        return collides((AAB) o);
    } else {
        throw new IllegalArgumentException();
    }
}
```
Double Dispatch

```java
interface Shape {
    collides(Shape o);
    collidesCircle(Circle c);
    collidesAAB(AAB aab);
}

public class Circle implements Shape {
    collides(Shape o) {
        return o.collidesCircle(this);
    }
    collidesCircle(Circle c) { /*code*/ }
    collidesAAB(AAB aab) { /*code*/ }
}

public class AAB implements Shape {
    collides(Shape o) {
        return o.collidesAAB(this);
    }
    collidesCircle(Circle c) { /*code*/ }
    collidesAAB(AAB aab) { /*code*/ }
}
```
Anonymous Methods

- Essentially an in-line class/interface
- All anonymous methods are inner classes
  - And therefore have a reference to the instance that creates them
Tips for Alc II

QUESTIONS?
What is a game designer?

• A game designer creates the experience and the “feel,” and is not just a programmer

• Games are a delivery system for your ideas
Compared to other media

• Designers of movies, books, and plays are creating a linear experience

• Interactivity is the defining feature of video games as a medium of entertainment
MDA Framework
Where to begin?

• Approach from the player’s perspective
  – What aesthetics do you want your game to have?
  – What do you want your players to feel?
• Create a basic idea that encapsulates those aesthetics
• Come up with dynamics that evoke the aesthetics
• First and foremost: know your audience!
  – Sunlab users? Competitive MOBA veterans? Kids?
  – Kids in the SunLab playing competitive MOBAs?
Further References

• Extra Credits: Playing Like a Designer
  – https://www.youtube.com/watch?v=HmtmoGwpZc
  – https://www.youtube.com/watch?v=QKEzMz6FcXs

• MDA: A Formal Approach to Game Design
  – http://www.cs.northwestern.edu/~hunicke/MDA.pdf
Bartle’s Taxonomy of Player Types

- **Killers**: Acting
- **Achievers**: World
- **Socializers**: Players
- **Explorers**: Interacting
Further References

• Extra Credits: Bartle’s Taxonomy
  – https://www.youtube.com/watch?v=yxpW2ltDNow

• Gamasutra (articles and news about game dev)
  – http://www.gamasutra.com
How to become a better designer?

• At Brown…
  – Make games, play games, come to BRGD
• Play lots of games!
Alchemy 1 Playtesting
Yay!