LECTURE 12
Announcements
This is the last lecture!

• Email the TA list if you need more help
• Next week is final presentations during normal class time
• Super deadline day shortly after (12/17)
  – Might want to hand in everything earlier…
Playtesting Reminders

- Don’t give hints or instructions
- Watch your playtesters: more useful than their feedback
- Turn in handwritten signatures
Deadline Approaching

• Course policy: you must turn in a working version of the engine for all projects
• Deadline for incomplete projects is December 17th at 11:59:59pm
  • Probably shouldn’t wait until then – we will determine final grades shortly after
  • Email the TA list when you hand in if you want faster feedback!
• Same day as Final V
QUESTIONS?
LECTURE 12
Collision Detection III
Point of Collision

• We want to find out where shapes hit
• In our simulation, colliding shapes are intersecting
  – Generally the intersection is small
  – So we choose a point that approximately represents where the intersection is
Poly-Poly

- When two polygons (AABs are polygons too!) collide, at least one vertex of one shape is inside the other (almost always)
  - If there’s only one point, use that as the point of collision
  - If there’s more than one, average them!
Circle-Circle

• Circle-Circle is easy:
  – It’s on the line connecting the centers, with ratio of the radii

• \[ \mathbf{p} = \mathbf{c}_1 + \frac{r_1}{r_1 + r_2} (\mathbf{c}_2 - \mathbf{c}_1) \]

• Remember this is in world (absolute) coordinates
Circle-Poly

• If vertices of the poly are within the circle, then average them
• If not, then take the point along the MTV:
  \[ \hat{p} = \hat{c} \pm r \hat{mtv} \]
• (Depends on MTV direction)
QUESTIONS?

Collision Detection III
Rotation

- We currently have shapes that don’t rotate
- First step is to be able to rotate shapes
- Next step is to provide collision response for rotating entities
Terminology

- Let's define some things:
  - Angle, $\theta$ (CCW)
  - Angular velocity, $\omega$
  - Angular acceleration, $\alpha$
  - Moment of Inertia, $I$
    - Analogous to mass (inertia) for rotation
Basics

- Your physical entities should have an angle, angular velocity, and angular acceleration.
- You should integrate these as before.
- But whenever you do this, you have to physically rotate the shape.

```java
public class PhysicalEntity{
    float angle, aVel, aAcc;
    
    void move(float time) {
        //integrate position
        aVel += aAcc*time;
        angle += aVel*time;
        aAcc = 0;
        rotate(aVel*time);
    }
}
```
Rotating Shapes

• What shapes do we need to rotate?
• AAB doesn’t rotate, by definition
• Circles are circles
  – You still need angular values for the circle though, what if a hitbox is a circle?
• Therefore only polygons need to rotate
• Rotate polygons by rotating their vertices
Centroid of Polygon

- Every shape rotates around its centroid
- The centroid of a polygon with $n$ vertices is:
  $$C_x = \frac{1}{6A} \sum_{i=0}^{n-1} (x_i + x_{i+1})(v_i \times v_{i+1})$$
  $$C_y = \frac{1}{6A} \sum_{i=0}^{n-1} (y_i + y_{i+1})(v_i \times v_{i+1})$$
  Where $A = \frac{1}{2} \sum_{i=0}^{n-1} (v_i \times v_{i+1})$
- $x$ and $y$ are coordinates of vertices in CCW order
Rotating Polygons

• To rotate a polygon, rotate each vertex by the angle $\theta$
  
  • $v_i'_{x} = v_{i_{x}} \cos \theta - v_{i_{y}} \sin \theta$
  
  • $v_i'_{y} = v_{i_{x}} \sin \theta + v_{i_{y}} \cos \theta$

• These vectors are the vertices relative to the centroid!

• Remember to update edges as well
Inertia

- We also need the moment of inertia of an object.
- You can define or calculate it.
- Circle: $I = \frac{1}{2} Mr^2$
- Polygon:
  
  \[
  I = \frac{M}{6} \sum_{0}^{n-1} (||\vec{v}_i||^2 + \vec{v}_i \cdot \vec{v}_{i+1} + ||\vec{v}_{i+1}||^2) (\vec{v}_i \times \vec{v}_{i+1}) \\
  \sum_{0}^{n-1} (\vec{v}_i \times \vec{v}_{i+1})
  \]
Physics III

ROTATIONAL PHYSICS
Impulse and Forces

- How can we cause shapes to rotate in the world?
- Currently we are applying impulse/forces the centroids of entities
- Apply impulse/force to object, but not at centroid
Impulse and Forces

• Now your impulses and forces have a magnitude and a point of application

• $\vec{r}$ is relative to the centroid

• The magnitude $\vec{i}$ is actually a vector

• $i$ for impulse and $f$ for force from now on
Angular Impulse and Forces

- $i_{\text{angular}} = \vec{r} \times \vec{i}$
- $f_{\text{angular}} = \vec{r} \times \vec{f}$
- In relation with angular velocity and acceleration:
  - $\Delta \omega = \frac{i_{\text{angular}}}{I} = \frac{\vec{r} \times \vec{i}}{I}$
  - $\Delta \alpha = \frac{f_{\text{angular}}}{I} = \frac{\vec{r} \times \vec{f}}{I}$
Collision Response

- We need to change the impulse we calculated in Physics II
- It’s now a different value that is applied at some specific point
  - It’s applied to the point of collision!
Some Definitions

• More definitions:
• $\vec{r}_a, \vec{r}_b$ are the vectors from the centroids of the shapes to the collision point
• $\vec{r}_{a\perp}, \vec{r}_{b\perp}$ are the perpendiculans to $\vec{r}_a, \vec{r}_b$
• $\hat{n}$ is the normalized MTV
Collision Response

- **Magnitude of the impulse**

\[ i = \frac{-(1+\text{COR})(u_a-u_b)}{\frac{1}{m_a} + \frac{1}{m_b} + \frac{(\vec{r}_a \cdot \hat{n})^2}{I_a} + \frac{(\vec{r}_b \cdot \hat{n})^2}{I_b}} \]

- \( u_a, u_b \) are projections of velocities onto the \( \hat{n} \)

- The impulse is in the direction of \( \hat{n} \), determine the sign based on your MTV direction
Fixed Rotation

• Just like with static shapes, there should also be shapes that don’t rotate
• Just like with the previous impulse equation, have a special case for non-rotating objects
  \[ \frac{1}{I_a} \] with 0 if the entity \( a \) doesn’t rotate
• Note that if both objects don’t rotate, the equation reduces to the old equation
QUESTIONS?
Physics III

FRICTION
Friction

- We don’t want everything to be slippery
  - Friction slows things down
- Give every physical entity a friction value greater than 0
- $COF = \sqrt{f_1 f_2}$
Frictional Force

• The frictional force is parallel to the surface of contact
  — i.e. perpendicular to MTV
• The direction is determined by the direction of the relative velocity (1D):
• \( u_{rel} = u_b - u_a \)
Relative Velocity

- Only velocity perpendicular to the MTV is relevant
- \[ u_{rel} = \vec{u}_b \cdot \hat{n}_\perp - \vec{u}_a \cdot \hat{n}_\perp \]
- Direction of the perpendicular (\( \hat{n}_\perp \)) doesn't matter
  - Consistency matters
How Much Force?

• From physics, the friction force on object A due to object B is proportional to the force exerted on object A by object B
• We don’t really have that force…
  – But we did apply impulse to the objects!
The Force

• So we have

\[ \vec{f} = C \| \vec{i} \| \text{sign}(u_{rel}) \hat{n}_\perp \]

• \( \vec{i} \) is the impulse applied in collision response

• \( C = k \text{ COF} \)

• \( k \) is a constant
Disclaimer

• This friction works for the case when the relative velocity is linear
• With rotation, things become much more difficult
• If you want to combine these, good luck!
LECTURE 12
Spacial Acceleration Structures
Collisions aren’t cheap

• An individual collision calculation is cheap...
• But number of collisions calculated is $O(n^2)$
  – 2 objects = 1 calc
  – 3 objects = 3 calcs
  – ...
  – $n$ objects = $\frac{n(n-1)}{2}$ calcs
• What if your world has 1,000 entities? 10,000? 1,000,000?
Can we do better?

- We can tell that some collisions don’t even need to be checked.
- Spatial acceleration data structures reduce the number of collisions by taking advantage of spatial locality.
  - Use as a replacement for your `Set/List<PhysicsEntity>`
  - Can even implement Java `Collection<T>`!
Bounding boxes

- Every shape has a bounding box – min/max x/y values of any point in the shape
- Function of the shape:
  - AAB – itself
  - Circle – center ± (r,r)
  - Poly – min/max over all vertices
  - Compound – min/max over all subshapes
Quadtrees

• Like a binary tree, but 4 children
• Divide world into quadrants
  – Recursively subdivide quadrants based on # of entities in a quadrant
  – Only try colliding with entities in your quadrant
• Demo: http://www.mikechambers.com/blog/2011/03/21/javascript-quadtree-implementation/
Quadtrees - insert

• Starting at the root node:
  – If this node isn’t split yet, add the entity here
    • If adding the entity results in X entities, split and re-insert everything in the split quadrant
  – Else, find the quadrant that should bound the entity’s bounding box
    • If one does, recur down that node
    • If none do, add the entity to this node
Quadtrees - retrieve

• Gets all the entities for possible collisions

• Starting at the root node: with an accumulator
  – Add all entities in this node to accumulator
  – Again, determine which quadrant this entity should reside in
    • If none, recur down all children – any entity could be valid!
    • Else, recur down that child quadrant
Quadtrees – pros/cons

• Pros
  – Fairly easy to implement
  – Good at reducing # of collisions

• Cons
  – Only works with bounded worlds
  – Assumes objects are uniformly distributed
Can we do even better?

- What if objects are highly concentrated (non-uniformly distributed)?
- What if the world is unbounded? Where to mark quadrants?
KD-trees

- K-dimensional trees (in our case, \( k = 2 \))
- Traversing to children is subdividing by either \( x \) or \( y \)
  - At each subdivision, choose the optimal \( x \) or \( y \) value to subdivide by
  - Efficiently splits up entities into buckets
Choosing the split axis

Split at middle?
Choosing the split axis

Split at median?
Choosing the split axis

Cost-Optimized Split
KD-trees – add/retrieve

• Implementation details are the same as quadtrees, except for splitting/traversal
  – Splitting – each node is an x or y node, subdivide based on coordinate at mean/median/etc..
  – Traversal – traverse based on x or y value – left is less than, right is greater than

• Subdivision may involve sorting – how to optimize # of sorts required?
KD-trees – pros/cons

• Pros
  – Takes entity distribution into account
  – Great at reducing # of collisions

• Cons
  – Trickier to implement
  – Slightly higher construction time
Using KD/quad trees

- On each tick:
  - Rebuild the tree (clear and then re-insert all entities)
  - Use retrieve(PhysicsEntity) to get entities to attempt collision with for each entity
  - Run standard collision detection algorithm
  - ???
  - Profit!
Spatial Acceleration Structures

QUESTIONS?
Lecture 12
Engines in the Wild
RPG Maker

- Spanned multiple platforms, but the main series is on Windows
- Has premade resources for easier accessibility
- Main gameplay elements already made for you
  - Focus more on story/puzzles than game mechanics
RPG Maker

- Simplified game entity language
  - Basic programming flow
  - Select various commands to execute from an option menu
- Simplified game entity language
  - Lots of things are already built in for you
  - People focus more on style and conventions of how you design the game
Map Examples
Game Maker: Studio

- Targeted for novice and professional developers
  - Great community support
- Cross-platform
  - Windows, OS X, iOS, Android, Ubuntu, Windows 8 Phone
- Included editors: sprite, sound, level, etc.
- Good example of a complete 2D game engine solution
Game Maker: Studio

- **Sprites**
  - sprite0

- **Sounds**
  - sound0

- **Backgrounds**
  - MyBackground

- **Paths**
  - path0

- **Scripts**
  - script0

- **Fonts**
  - font0

- **Time Lines**
  - timeline0

- **Objects**
  - object0

- **Rooms**
  - room2
  - room0
  - room1

- **Game Information**
- **Global Game Settings**
- **Extension Packages**
Game Maker: Studio

- Rapid prototyping
- Drag ‘n’ drop commands
- GML scripting language
  - Imagine weak-typed C
Game Maker: Studio
Game Maker: Studio
Game Maker: Studio
Game Maker: Studio
Game Maker: Studio

- Cross-platform engine targeted for novice and professional developers
- Drag’n drop commands
- GML scripting language
  - Imagine weak-typed C
- Included editors: sprite, sound, level, etc.
- Good example of a complete 2D game engine solution
Engines in the Wild

FRAMEWORKS/LIBRARIES
Slick2D

• Java game engine build on-top LWJGL
• Essentially a wrapper for LWJGL expanding on:
  – Application/Screen
  – Timed and fixed update method (i.e. onTick())
  – Room transitions
• Removes the need for OpenGL calls
• No physics, artificial intelligence, networking
Kryonet

- Abstracts out low-level socket calls.
- Essentially sends messages in a serialized class defined by you (think of classes as structs)
- Recommended if you want to avoid Java Sockets and focus on network design
Box2D

- Popular C++ 2D physics engine
- Faster and more stable than our physics engines
  - Continuous collision detection
- Friction, rotation, joints
- Can be used for fluids and particles
Tips for Final IV and V

JAVA TIP OF THE WEEK
The Many Uses of `final`

- Did you know? `final` can be applied to:
  - Instance variables
  - Local variables
  - Method parameters
  - Classes
  - Methods
Final Instance Variables

• Value never changes
• Can be set in constructor or initializer block
  – Must be set by the time an instance is created
• Can be different between instances of same class
• Remember, doesn’t make an object immutable!
• Usage: good for public fields and constants

```java
class Example {
    private final float mass;
    private final String name;
    private final int[] values = {1, 2, 3, 4, 5};

    public Example(String name, float mass) {
        this.name = name;
        this.mass = mass;
    }
}
```
Final Local Variables

- Must be set immediately at declaration
- Value can’t change
- Usage: give anonymous classes access to local variables
  - This way the value can be passed as an implied argument, but this strategy doesn’t work if the value will change

```java
public void addShowField(String text) {
    final TextField field;
    field = new TextField(text);
    field.setVisible(false);
    this.add(field);
    Button button = new Button("Click to show",
        new ButtonListener() {
            public void onClicked() {
                field.setVisible(true);
            }
        });
    this.add(button);
}
```
Final Parameters

• Special kind of local variable, same behavior
• Set by caller, can’t change once in method
• As Java always passes by reference, this does not affect the caller
• Usage: very little
  — However, mutating parameters is generally bad practice, so it doesn’t hurt to mark them final

```java
public boolean contains(final String query, final int start, final int end) {
    //illegal
    while (start < end) {
        start++; …
    }
    //legal
    for (int i = start; i < end; i++) {
        …
    }
}
if (sequence.contains("stuff", 0, 5))…
if (sequence.contains("things", 8, 60))…
```
Final Classes

- Can’t be extended
- Often seen in libraries or other places where class hierarchy may not be clear
- Usage: prevent other programmers from breaking things by overriding key functionality

```java
public final class Data {
    private int count;
    private float total;
    ...
    public float getAverage() {
        return total / count;
    }
    public void add(float datum) {
        total += datum;
        count++;
    }
}
```
Final Methods

- Can’t be overridden
- Selectively allow inheritance
- In a final class, all methods are final
- Usage: now there’s no need to rely on subclasses to call super

```java
class MyClass {
    // can’t be overridden
    public final void init() {
        // run some common init routine
        internalInit();
    }

    // can be overridden
    // perform subclass-specific init
    protected abstract void internalInit();
}
```
QUESTIONS?
CS1972: Topics in 3D Game Engine Development
Presentation format

• 5-10 minutes long
• Any number of slides in .ppt[x] or multiple .pngs
  – But be sure to keep it in the time range
  – Make sure to have a copy ready for class
• Distribute speaking evenly among teammates
• Order will be picked randomly as we go!
  – Don’t be late!
• Must be a postmortem!
Postmortems

• 5 things that went well, 5 that didn’t
• Purpose is to think about what you should and shouldn’t do next time you make a game (or any large project, really)
• Can be about anything related to development: team dynamics, decisions about code/content
Screen capture

• To record a video on the department, you can run (on one line):
  
  ```
  avconv -f x11grab -r 24 -s 1280x720 -i DISPLAY
  -vcodec libx264 -pre lossless_ultrafast -threads 0 /ltmp/video.mkv
  ```

  Records video that is 1280x720 pixels starting at position (X,Y) on screen DISPLAY
  
  Get display with command `echo $DISPLAY`
  
  — May take a couple of tries to get position right

• Outputting to /ltmp recommended
Goodbye

• Thank you for taking this course.
• We hope you enjoyed the ride
  — The TAs did!
• The work you have done has been amazing
• Apply to HTA and TA 195n next fall!
Final III Playtesting!

Last in-class playtest! :'( 