Lecture 6

Announcements

Mid-Semester Feedback

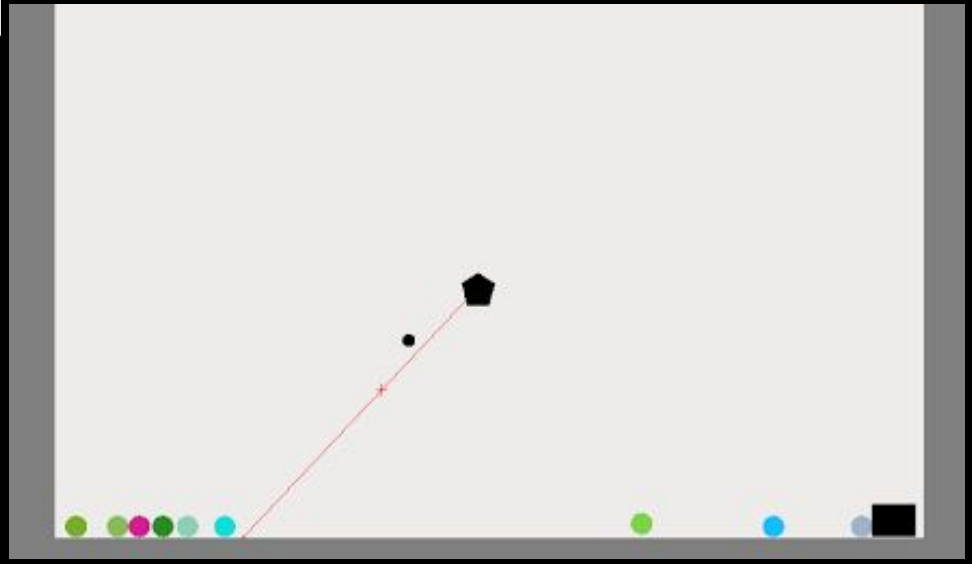
- Please fill out the feedback form!!
- You can do it today after playtesting!!!

Special Topics Lectures

- This is the last week we're doing normal lectures
- Starting next week, we'll be introducing various things you can add to your final project
  - Possibly featuring BRGD!

This Week: Nin2

- A real game, not just a physics demo!
- Make sure to come to hours if you're spending huge amounts of time debugging
  - This week's topics can be a bit tricky



Nin1 Initial Feedback

- Gravity is a force, not an impulse
  - If you set it as an impulse, fall speed is dependent on frame rate
- The gravitational constant (“g”) is an acceleration, not a force
  - Multiply by mass (F = ma)
  - Otherwise, heavy objects fall slower than light objects
- Play around with g- if things are falling too slowly, crank it up

Nin1 Initial Feedback

- Are your players warping through platforms?
- Try setting a maximum tick duration!
  - If tickLen > maxLen:

    do multiple ticks of length maxLen

Nin1 Initial Feedback

- Help! Objects are sinking into each other when they collide!
  - Remember to move the objects away by half the MTV (or by 1 – mass ratio)

Nin1 Initial Feedback

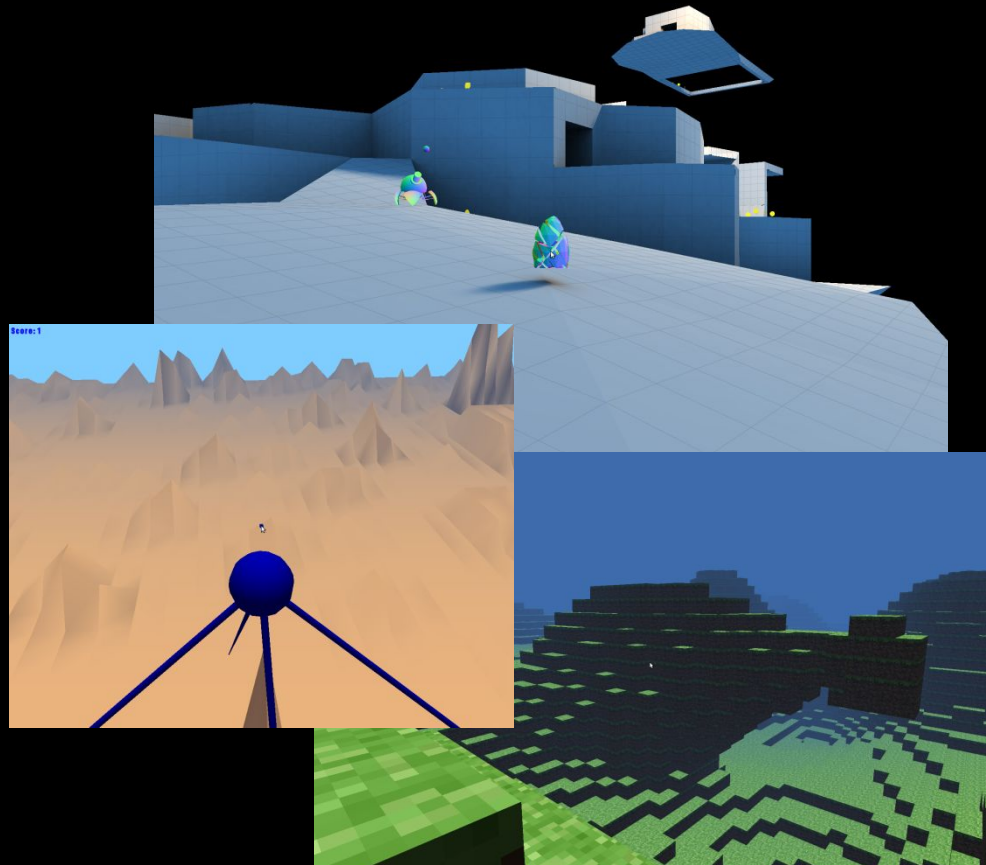
- Help! Objects are sinking into each other when I stack them!
  - Run collision resolution multiple times per physics step until stabilization
  - Make sure to cap the number of checks per tick

Final Design

- Meeting with the TA staff to talk about your final project
  - Next week after lecture
  - Email us engine ideas by this Friday
- You'll be telling us:
  - What engine features you'll implement (per person)
  - Whose engine(s) you'll build off of
  - How you will use version control

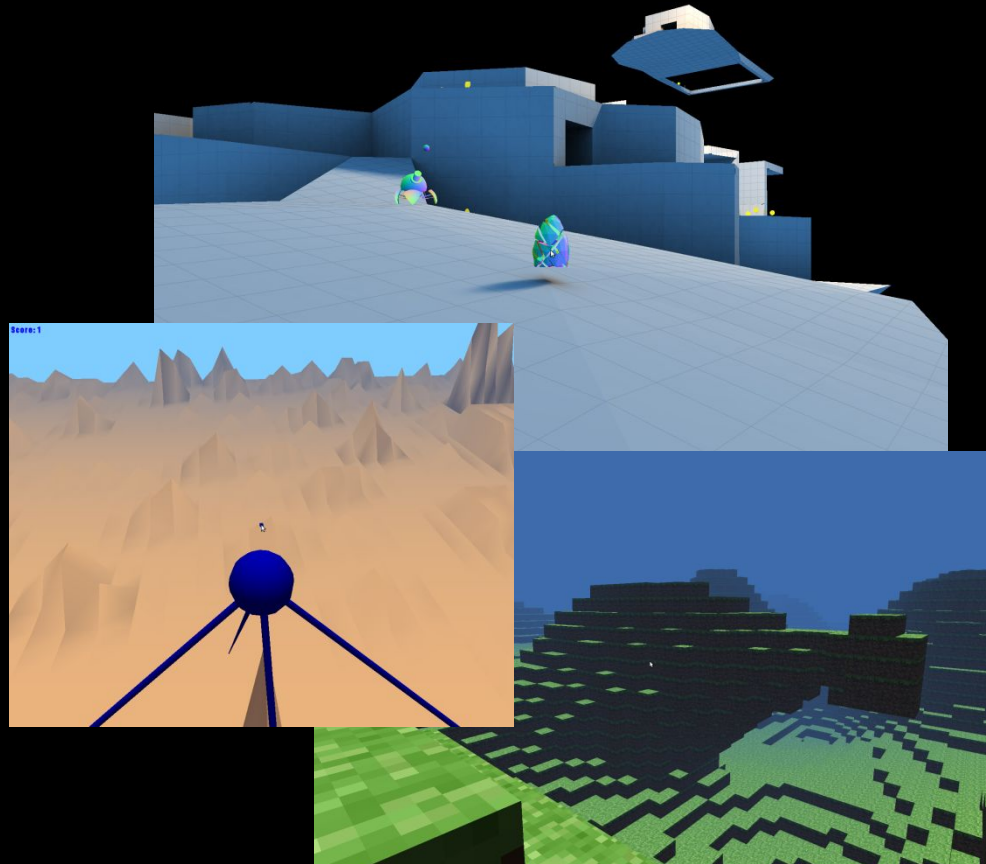
CS195u: 3D Game Engines

- Running next semester!
  - 12p-1p Wednesday in 316
- Two (soft) prerequisites:
  - Software engineering: 1971, 32, or 33
  - Graphics: 123
- Topics include physics, world/level representation, pathfinding over navigation meshes
- cs.brown.edu/courses/csci195u/
- See the website for more details
- You can run the project demos in /course/cs1972/demo



CS195u: 3D Game Engines

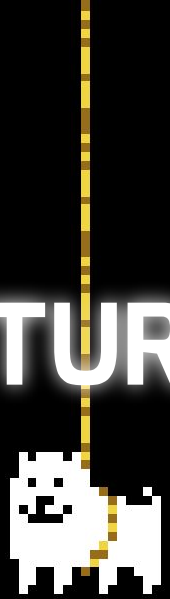
- cog
- sphere
- adrenaline
- roam
- dragonfly
- Administrator
- PolyhedroneDefense
- castle_defense
- mystic
- lifesimulator2017
- ampli



Announcements

QUESTIONS?

Lecture 6

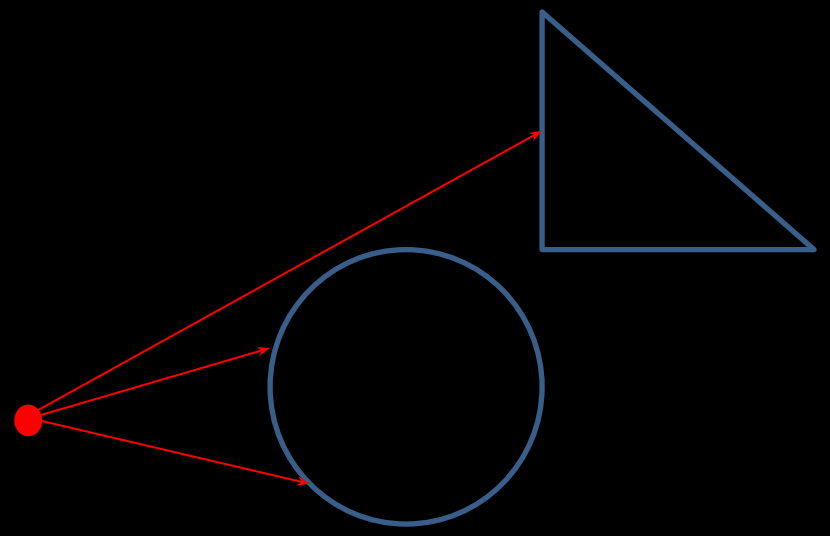


Hang in there!

Ray Casting

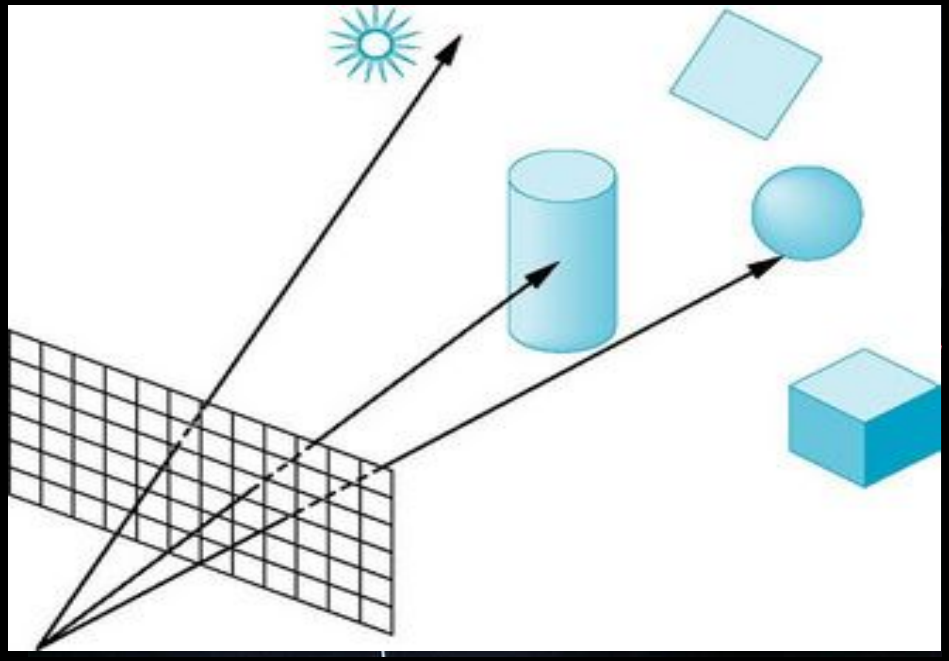
What is raycasting?

- Determine the first object that a ray hits
- A ray is like a ray of light, has a source and direction and goes on forever
- Think of it as shooting a laser in a particular direction



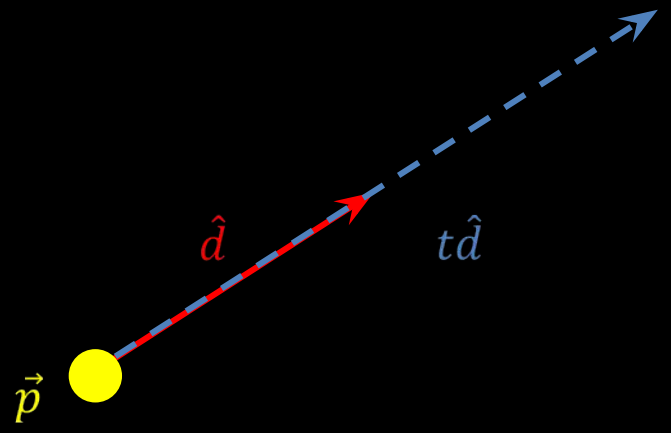
Raycasting Uses

- When would we need to raycast?
  - Hitscan weapons
  - Line of sight for AI
  - Area of effect



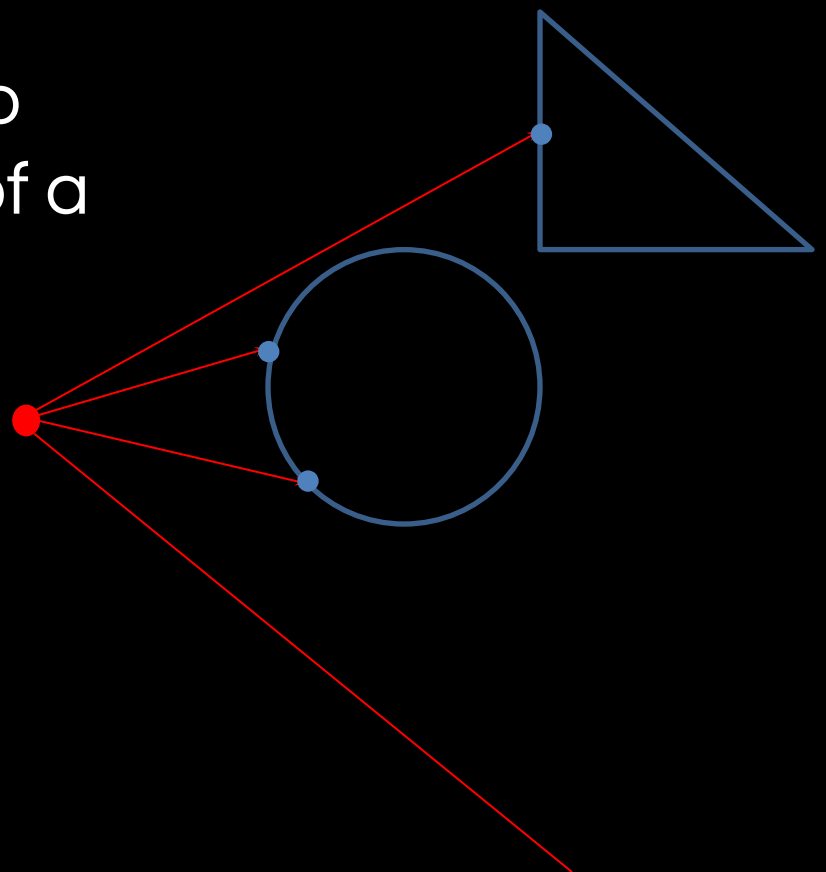
The Ray

- A ray is a point (source) and a direction
- Point on ray given by:
- $\vec{r} = \vec{p} + td$
- $\vec{p}$ is the source
- d is the direction
  - This must be normalized!
- t is a scalar value (length)



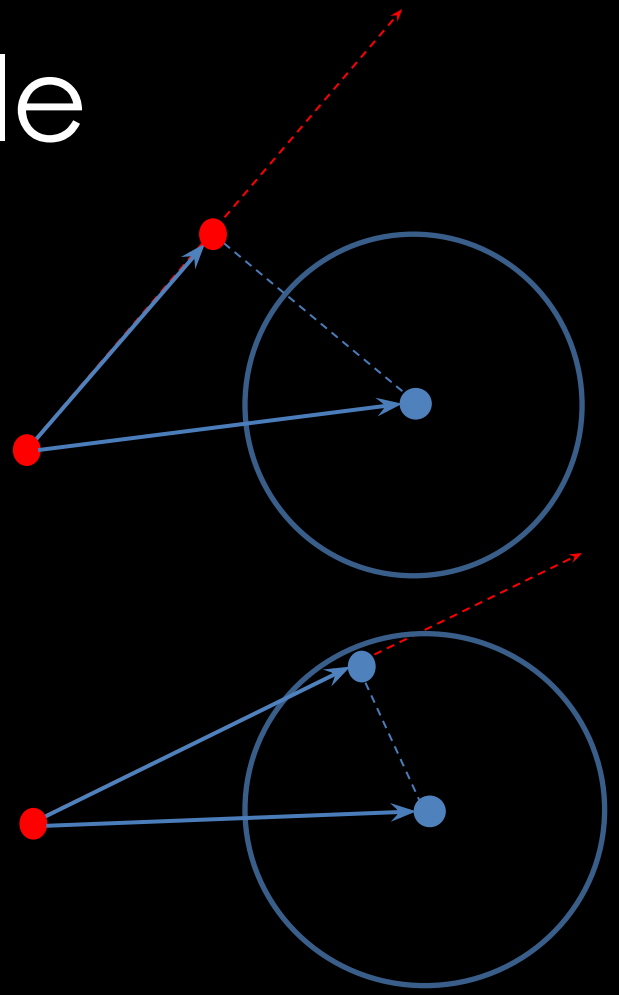
Basics

- Raycasting boils down to finding the intersection of a ray and shapes
- Kind of like collision detection all over again
- You want the point of collision as well



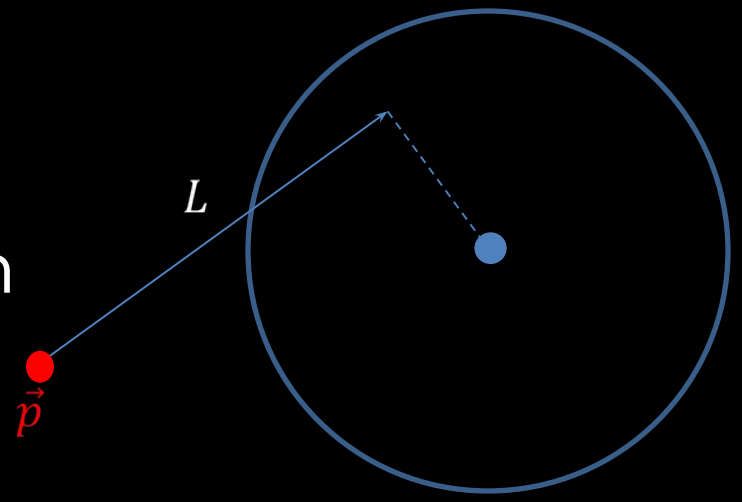
Ray-Circle

- If the source is outside
- Project center onto ray
- Check if the projection is positive and the projection point is within the circle



Ray-Circle

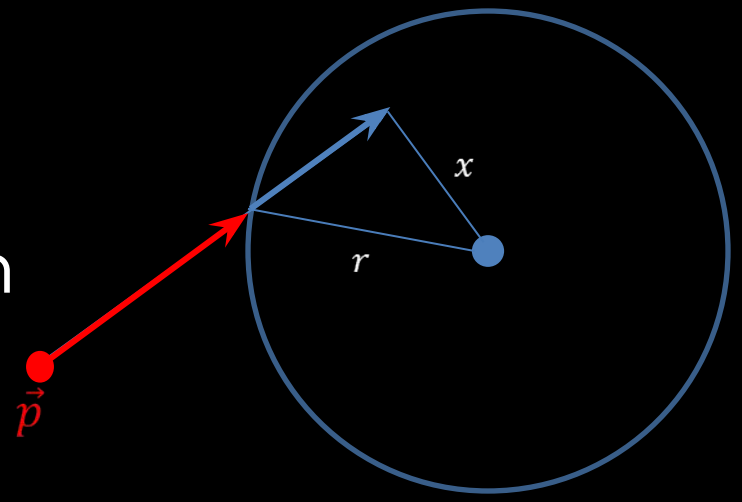
- If the source is outside
- Project center onto ray
- Check if the projection is positive and the projection point is within the circle
- Point of intersection?



Ray-Circle

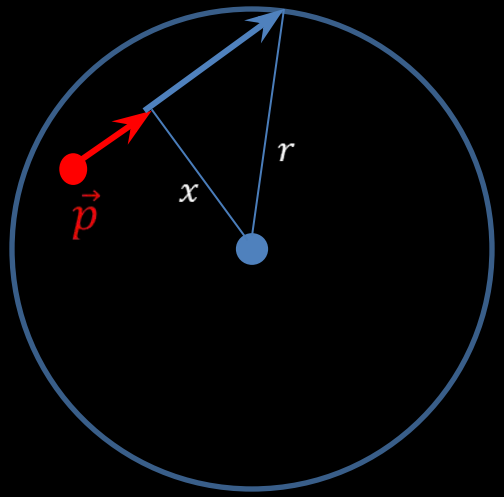
- If the source is **outside**
- Project center onto ray
- Check if the projection is positive and the projection point is within the circle

$$p + d\left(L - \sqrt{(r^2 - x^2)}\right)$$



Ray-Circle

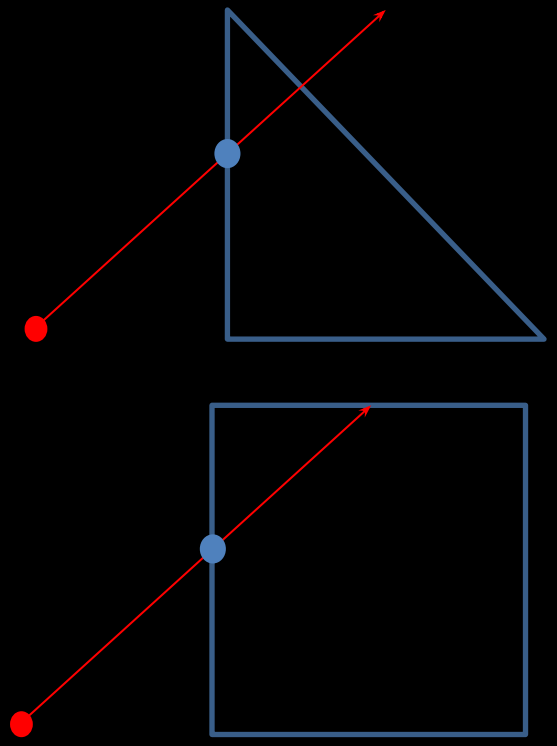
- If the source is **inside**
- Project center onto ray
- Projection must be in the circle
- Projection can be negative



$$p + d\left(L + \sqrt{(r^2 - x^2)}\right)$$

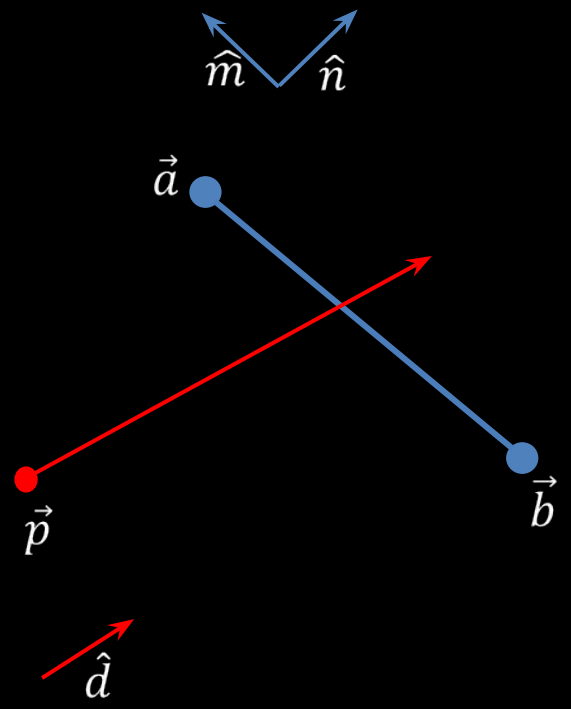
Ray-Polygon/AAB

- A polygon/AAB is composed of edges
- We can check for intersection of ray by checking for intersection of all edges
- There is no shortcut for AABs this time



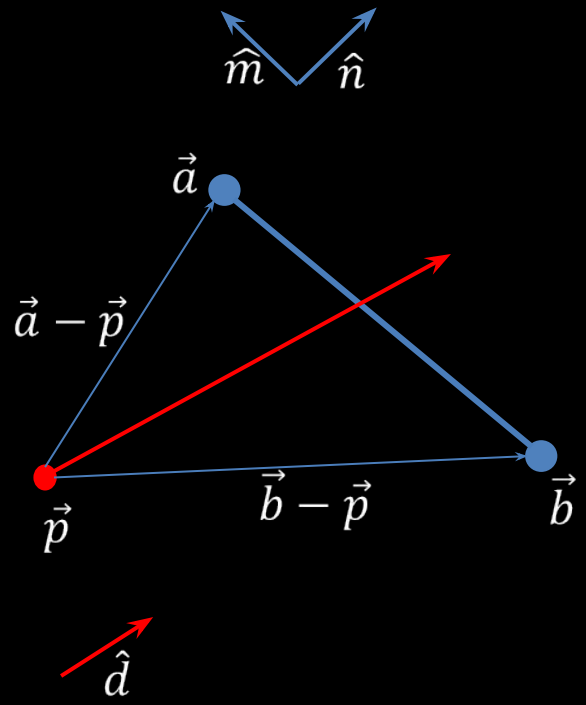
Ray-Edge

- Edge is defined by two end points, $\vec{a}$ and $\vec{b}$
- We need some other vectors:
- m is direction of the segment (normalized)
- n is the perpendicular to the segment (normalized)



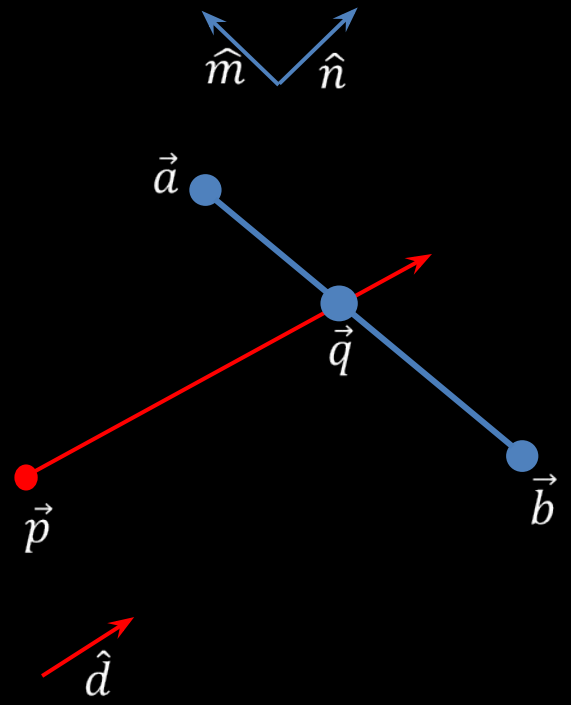
Ray-Edge

- Firstly, determine if the segment straddles the ray
- Use cross products
- We have support code for this
- $(\vec{a} - \vec{p}) \times \vec{d}$ and $(\vec{b} - \vec{p}) \times \vec{d}$ must be of opposite sign
- If the product of the two cross products is greater than 0, there is no intersection



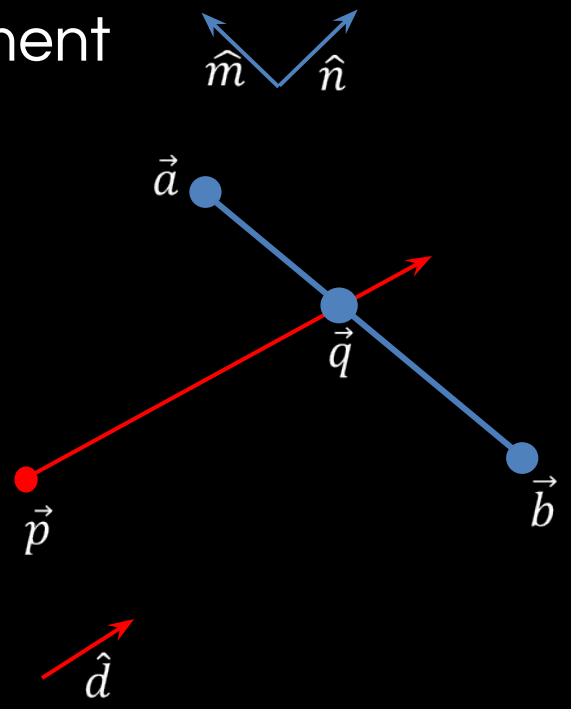
Ray-Edge

- Secondly, determine where the two lines intersect
- Point of intersection
  - $\vec{q} = \vec{p} + t\vec{d}$
- Solve for t
- t must be nonnegative



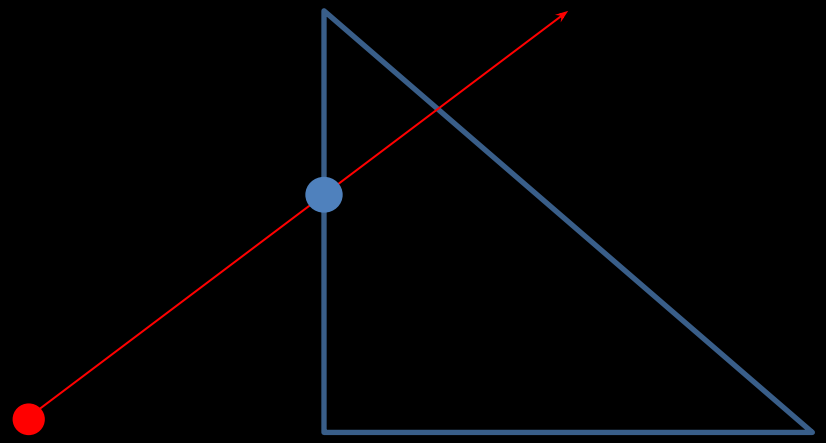
Ray-Edge

- Because $\vec{q} - \vec{b}$ lies on the segment
- $(\vec{q} - \vec{b}) \cdot n = 0$
- So plugging in:
- $(\vec{p} + td - \vec{b}) \cdot n = 0$
- $td \cdot n = (\vec{b} - \vec{p}) \cdot n$
- $t = \dfrac{(\vec{b} - \vec{p}) \cdot n}{d \cdot n}$



Ray-Polygon

- Intersect the ray with all the edges of the polygon
- Ray intersects polygon if it intersects at least one edge
- Keep track of the point closest to the source (lowest *t* value)



Putting it all together

Raycasting:

1. Intersect ray with every shape in the world
   - For circles, use the circle-ray algorithm in the slides
   - For polygons and AABs, intersect each edge and use the closest
2. Keep track of closest intersection point from the source as well as the corresponding shape

Raycasting

QUESTIONS?

Lecture 6

Saving/Loading

*It's possible that you might have a problem

Parsing Txt Files Can be Hard

- Must read 1 line at a time
- No easy lookups
- Poor formatting or inconsistent data could be anywhere!

Use XML!

- Java supports XML I/O!
- Can query the file for elements by name, ID, attribute, and more
- Information can be organized into hierarchies

```
<level id="lvl2.1">
    <player name="The_Hero" health="102" mana="50">
        <inventory>
            <sword quality="Dull"></sword>
            <armor quality="Rusty"></armor>
        </inventory>
    </player>

    <Enemies>
        <boss health="2000"></boss>
    </Enemies>
</level>
```

Writing XML

- XML looks like HTML
- XML has a very rigid structure
- Errors/typos will cause the parser to fail

Writing XML

- XML Declaration:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

- Must be at the top of each XML file

Writing XML

- Construct tags to hold information
- Each opening tag must match a closing tag

<Tag> ← opening tag

</Tag> ← closing tag

Writing XML

- You can nest tags
- Must close tags in the reverse order that they were opened

```
<OuterTag>
    <InnerTag>
    </InnerTag>
</OuterTag>
```

Writing XML

- Each pair of tags can hold an arbitrary number of inner tags
- You can freely reuse tag names

```
<Tag>
    <Tag></Tag>
    <Tag></Tag>
</Tag>
```

Writing XML

- Tags can be arbitrarily deep (as long as each one is closed)

```
<OutermostTag>
    <MiddleTag>
        <InnerTag>
            <EvenMoreInnerTag>
                ...
```

Writing XML

- Can put text in between innermost tags
- Can use numbers, but they're parsed as strings

```
<OuterTag>
    <InnerTag>text, ints, whatever</InnerTag>
    <AnotherTag>5</AnotherTag>
</OuterTag>
```

Writing XML

- Can add extra information to a tag (attributes)

```
<OuterTag name="outer">
    <InnerTag id="inner">
    </InnerTag>
</OuterTag>
```

Writing XML

- Tags can close themselves

```
<SelfClosingTag/>

<AnotherTag name="tag2"/>
```

Writing XML

- Comments are held between <!-- and -->
- Comments are multiline
- Very useful for commenting out parts of the file

```
<!-- I am a comment! -->
<RealTag></RealTag>
<!--CommentedTag></CommentedTag-->
```

Writing XML

- Each file **must** have exactly one pair of outermost tags
  - This doesn't include the XML declaration at the beginning

Writing XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<Game>
    <Map w=”5” h=”5”>01010000001101111000000101</Map>
    <Object id=”player” x=”42” y=”17”>
        <SpriteBehavior image=”player.png”/>
        <CollisionBehavior shape=”AAB” w=”10” h=”25”/>
        <!-- more behaviors -->
    </Object>
    <Object … >
        <!-- more behaviors -->
    </Object>
    <!-- more objects -->
</Game>
```

Reading XML

```
import javax.xml.parsers.DocumentBuilderFactory
import javax.xml.parsers.DocumentBuilder
import org3.w3c.dom.Document


// Setup the parser
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder docBuilder = factory.newDocumentBuilder();
Document doc = docBuilder.parse("<file_path>");
doc.getDocumentElement().normalize();
```

Reading XML

```
Node node = doc.getDocumentElement();
System.out.println(node.getNodeName());

for(Node n : doc.getElementsByTagName("Object")) {
    if(n.getNodeType() == Node.ELEMENT_NODE) {
        Element e = (Element) n;
        e.getAttribute("e");
        e.getElementsByTagName("SpriteBehavior");
        e.getChildNodes();
    }
}
```

Useful Classes

- DocumentBuilderFactory
- DocumentBuilder
- Document (org.w3c.dom NOT javax/swing)
- Element
- Node & NodeList

Reading an XML

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder docBuilder = factory.newDocumentBuilder();
Document doc = docBuilder.parse(“<file_path>”);
doc.getDocumentElement().normalize();

NodeList nList = doc.getElementsByTagName("Player");
Element player = (Element) nList.item(0);

System.out.println(player.getAttribute(“Health”));
System.out.println(player.getChildNodes());

--Output:--
>>>”100”
>>>[Element named “Sword”]
```

Writing XML

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder docBuilder = factory.newDocumentBuilder();
Document doc = docBuilder.newDocument();

// Create elements, and attributes to them
Element player = doc.createElement("Player");
player.setAttribute("Health", "100");

Element sword = doc.createElement("Sword");

// Add child elements to other elements, and top element to the doc
player.appendChild(sword);
doc.appendChild(player);
```

Saving and Loading

QUESTIONS?

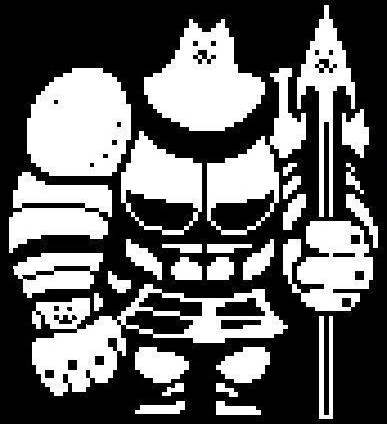
You loaded the file… now what?

- Keep a reference of the available Behavior types you have
  - `Map<String, Class<? extends Behavior>>`
- Keep a reference of the GameObjects in your level
  - `Map<String, GameObject>`
- `NodeList objList = doc.getElementsByTagName("object");`
- Iterate over each entry in the list and translate it into a `GameObject`

Initializing Behaviors

- For each 'object' element:
  - Make a new `GameObject`
  - Get the behaviors
    - `NodeList bList = obj.getElementsByTagName("Behavior");`
  - Iterate over them; initialize each behavior using your map of available behaviors
  - Add the behaviors to your new `GameObject`
- It's your engine; give each `Behavior` a special constructor or initializer!

Lecture 6

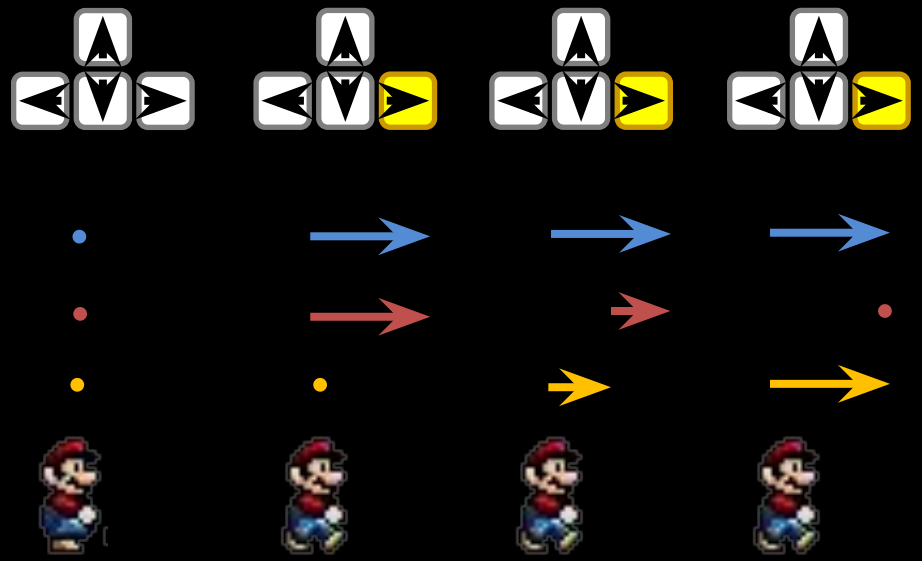


Greater Dog.

Tips for Nin II

Goal velocity

- `goalVelocity` set directly from arrow keys
- Gradually set `velocity` to `goalVelocity`
- By applying a force
  - $F = k(v_{goal} - v_{current})$



Constructing a world

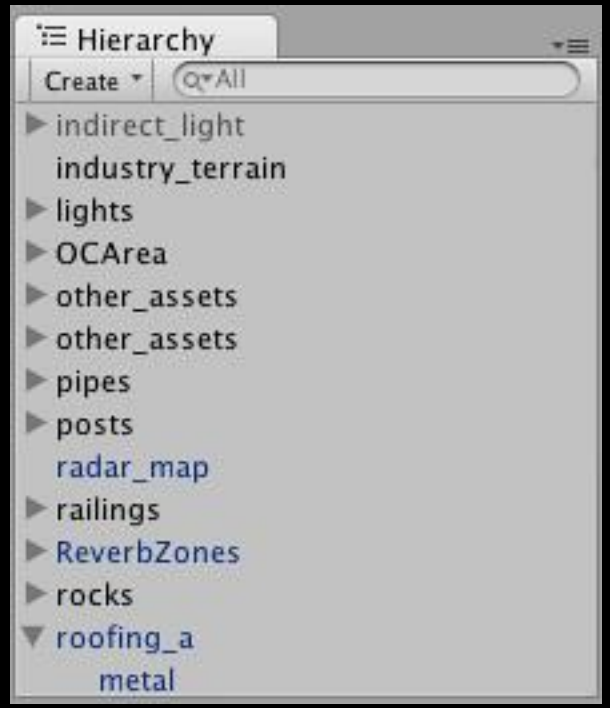
- We have new support code!
  - `CS1971LevelReader`
  - `LevelData`
- Use the properties of LevelData to populate your world

Constructing a world

- We have new support code!
  - `CS1971LevelReader`
  - `LevelData`
- Use the properties of LevelData to populate your world

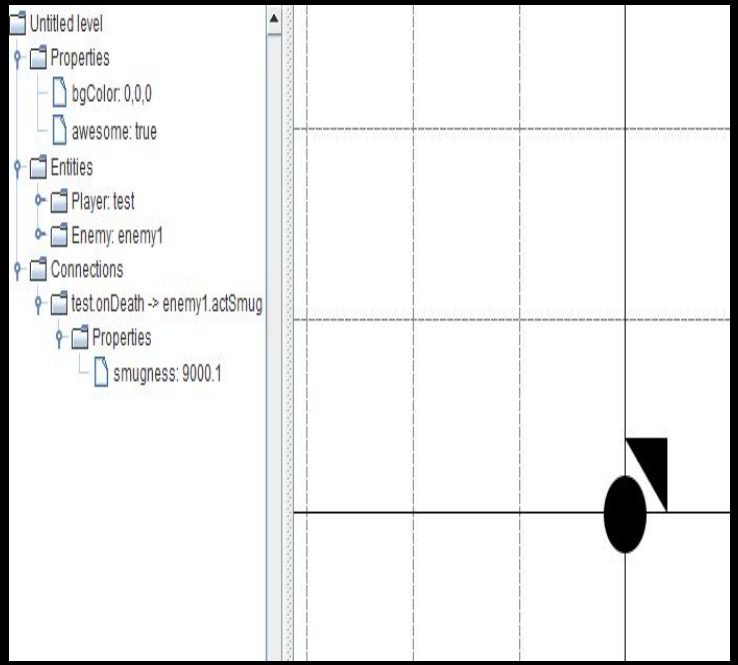
Constructing a world

- Keep a reference of the available classes/Behavior types you have
  - `Map<String, Class<?>>`
- Keep a reference of the GameObjects in your level
  - `Map<String, GameObject>`



Constructing a world

- Iterate over all the entities in your level object and translate that into actual `GameObjects`
  - 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 `Outputs` and `Inputs` 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

Tips for Nin II

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

```
// 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`

```
// 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

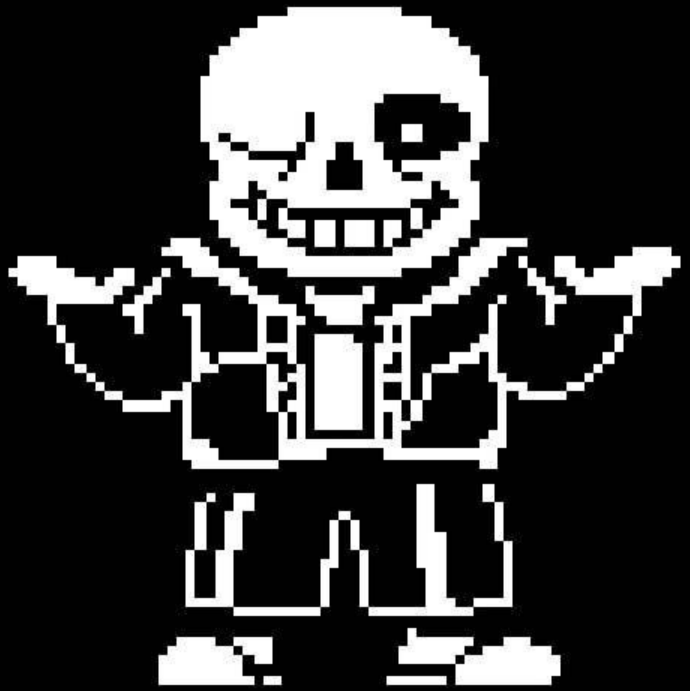
- Arbitrary blocks of code can be labeled
- Therefore you can have an arbitrary break
- Whee! It’s like a goto!
  - But don’t use it like one
  - Can only jump within the encapsulating block

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

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

Game Design

Story



Advantages of Story

- 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


Collect all 8 pages

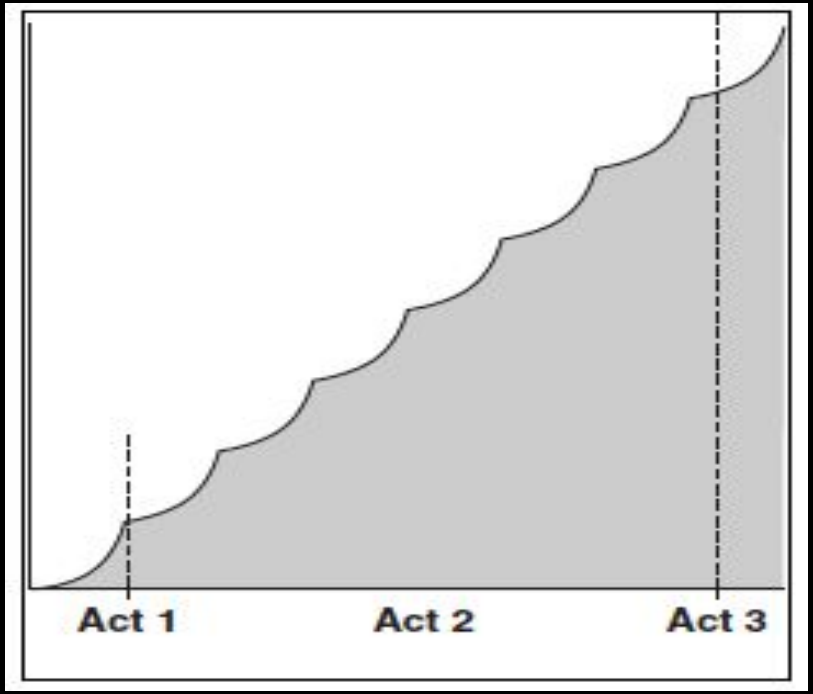
What to include?

- 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
    - http://www.kongregate.com/games/jordanmagnuson/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

USA

Deutschland

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*

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

Story in Games

QUESTIONS?

Nin 1 Playtesting

Let’s do it!