Principles of Computer Game Design and Implementation

Lecture 11

We already learned

- Vector operations
  - Sum
  - Subtraction
  - Dot product
  - Cross product
  - A few others about jMonkey, eg. User input, camera, etc

Outline for Today

- jMonkey Bits
- Collision detection – overlap test and intersection test

jMonkeEngine Bits and Bobs

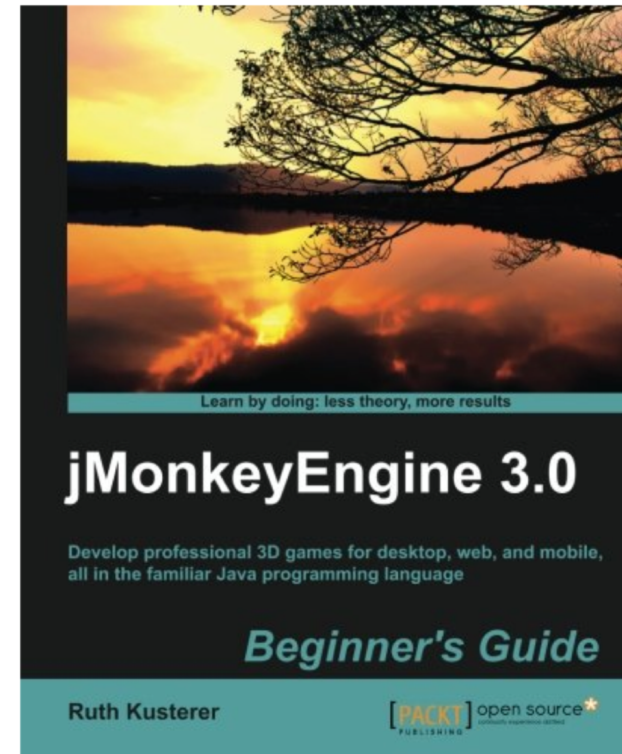
Computer Games…

- … are not just about graphics and entity manipulation. One needs (among other things)
  - Camera control
  - Keyboard input
  - Mouse events
  - Text info
  - Textures and materials
  - Audio

We are going to look at these issues

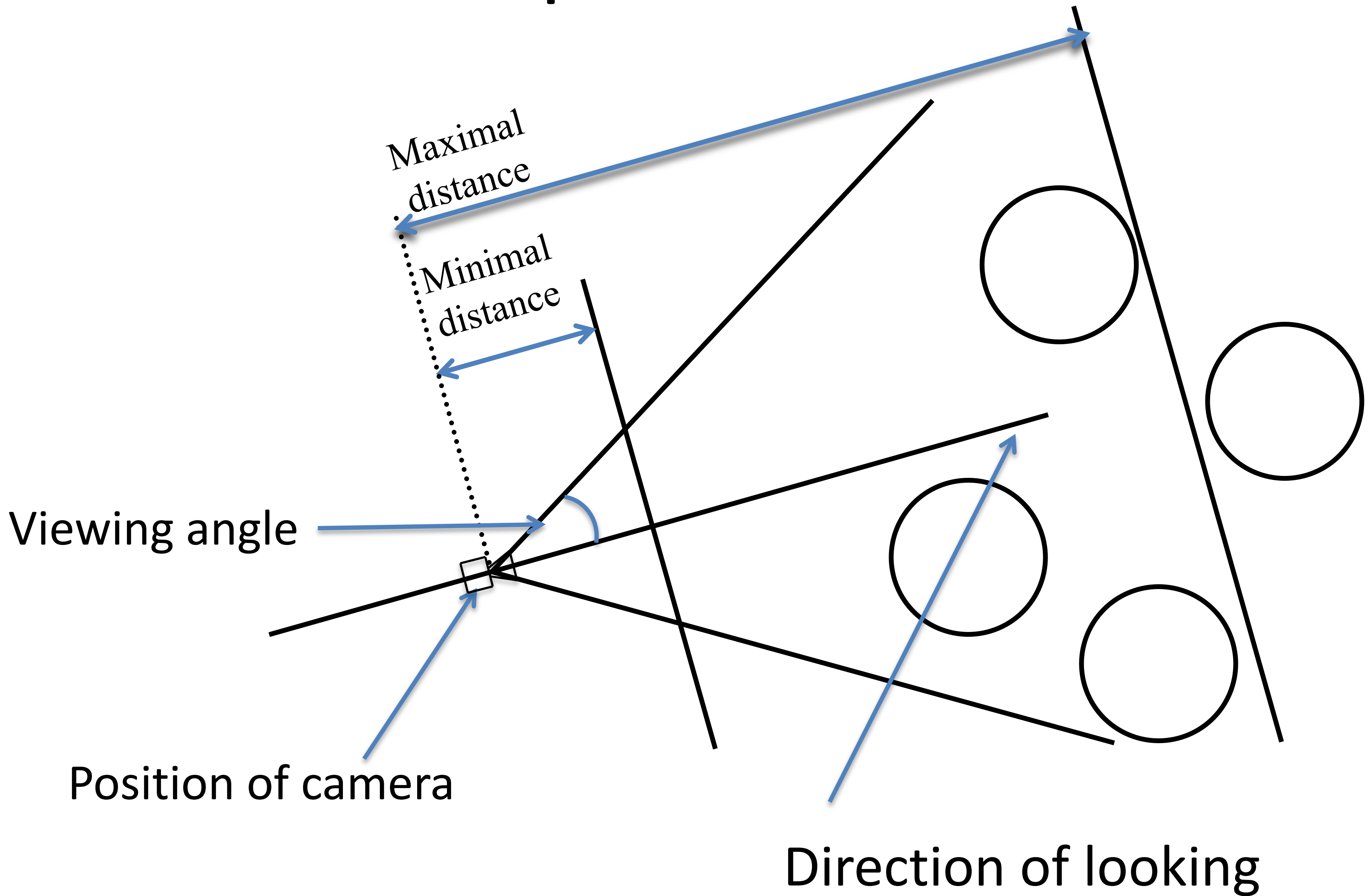
Just the Bare Minimum

- Much more information can be found on the jMonkeyEngine web site & in the Book



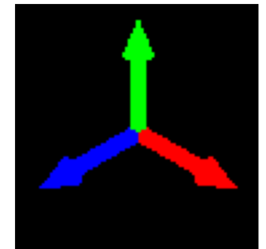
- Examples are based on jME tests and tutorials

Simple Camera



HelloCamera

```
cam.setFrustumPerspective(45.0f,
(float)settings.getWidth() /
(float)settings.getHeight(), 1f, 100f)
```



Viewing angle

Aspect ratio

Min distance

Max distance

```
cam.setLocation(new Vector3f(10, 10, 10));
```

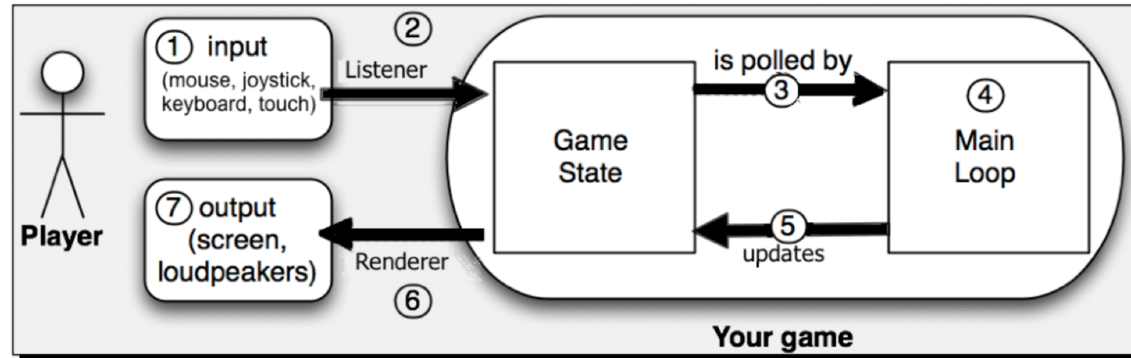
Camera location

```
cam.lookAt(Vector3f.ZERO,Vector3f.UNIT_Y);
```

Look at

Direction "Up"

Key & Mouse Bindings



- Events are mapped to triggers
- Triggers call action/analogue listeners
- Action/analogue listeners are called from the main loop

Example ActionListener

```
private ActionListener actionListener = new
ActionListener(){
  public void onAction(String name,
            boolean pressed, float tpf) {
    if(name.equals("Move right")){
      gBox.move(5*tpf,0,0);
    }
    else if(name.equals("Move left")) {
      gBox.move(-5*tpf,0,0);
    }
}
```

Sample AnalogListener

```
private AnalogListener analogListener = new
AnalogListener() {
  public void onAnalog(String name,
               float value, float tpf) {
  if(name.equals("Move right")){
    gBox.move(5*tpf,0,0);
  }
  else if(name.equals("Move left")) {
    gBox.move(-5*tpf,0,0);
 }
```

Deceleration

- We will look in more detail later, but for now
  - Simulate a slowing ball motion

HelloDeceleration

```
public class Example07 extends
SimpleApplication {
   Vector3f direction = new Vector3f(1,0,0);
   float speed = 5;
   Geometry gBox;
     … … …
   protected void simpleUpdate() {
        speed -= 2*tpf;
        if(speed < 0.01f) {
           speed = 0;
        }
 gBox.move(direction.mult(boxSpeed*tpf));   }
```

Direction of motion

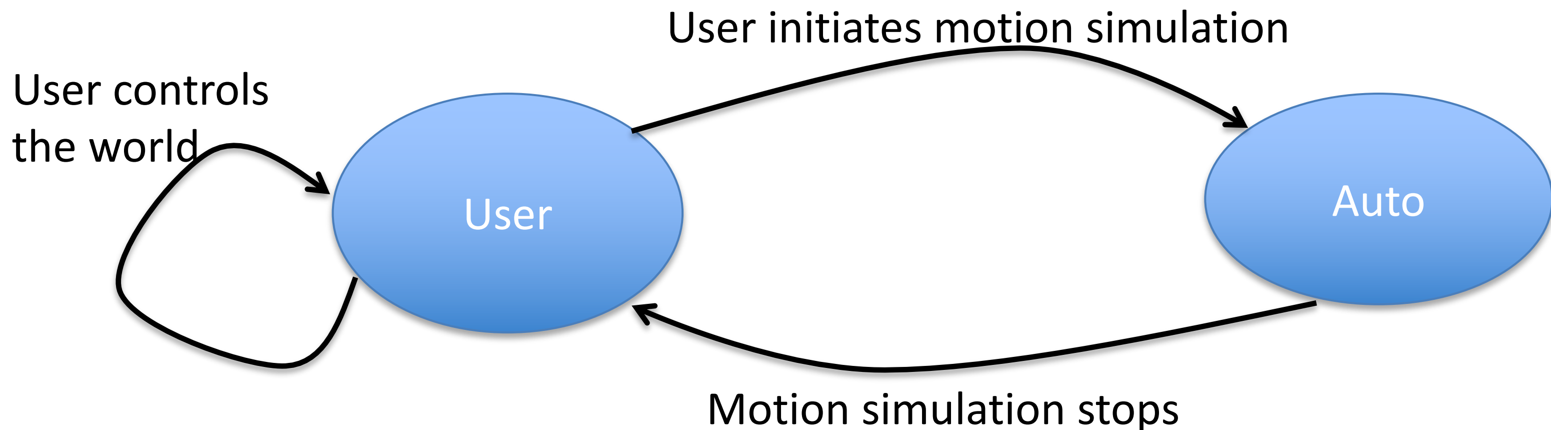
Velocity

Reduce the speed gradually

Make sure it zeroes

User Control V Modelling

- In these examples, user controlled completely the state of the world or there was no user input.
  – How to mix user control and physical modelling?
    - Game states



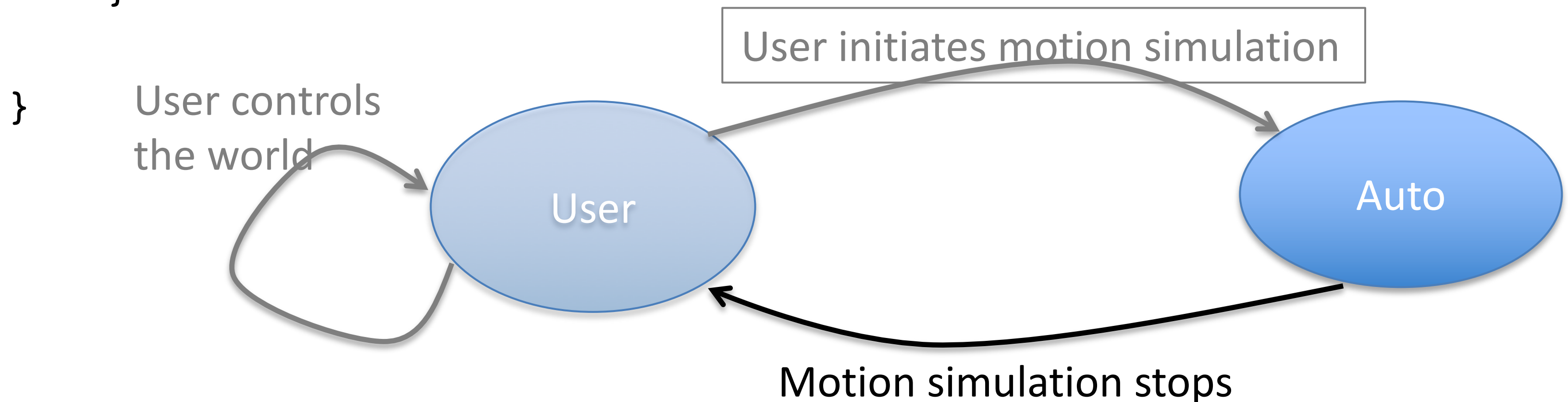
Game States

- jME3 provides *good* support for game states
- We use a simple `switch` operator
  - `enum State {user, auto};`
  - `State state = State.auto;`

simpleUpdate

```
public void simpleUpdate(float tpf) {
        switch(state) {
            case auto:
            boxSpeed -= 2*tpf;
            if(boxSpeed < 0.01f) {
                boxSpeed = 0;
                state = State.user;
            }
            gBox.move(direction.mult(boxSpeed*tpf));
        }

    }
```



onAction

```
public void onAction(String name, boolean isPressed, float tpf){
  switch(state) {
    case user:
      if(name.equals("Move right")){
        boxSpeed = 5;
        direction = new Vector3f(1,0,0);
        state = State.auto;
      }
      else if(name.equals("Move left")) {
        boxSpeed = 5;
        direction = new Vector3f(-1,0,0);
        state = State.auto;
      }
      break;
    case auto:
      // do nothing
  }
}
```

Text Fields

```
guiFont =
   assetManager.loadFont("Interface/Fonts/
                                Default.fnt");
BitmapText text = new BitmapText(guiFont);

text.setSize(guiFont.getCharSet().getRende
redSize());
text.move(settings.getWidth() / 2 + 50,
          text.getLineHeight() + 20,
          0);
text.setText("Ha ha ha!");
guiNode.attachChild(text);
```

Collisions

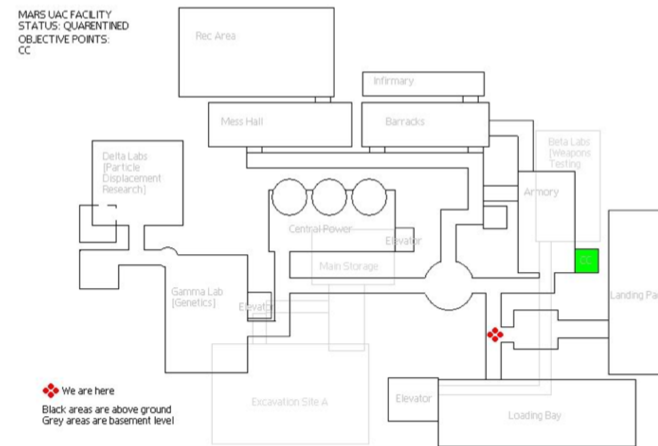
- Collision detection
  - Do moving entities collide?
  - Mostly geometry and algorithms
- Collision response
  - How to react to a collision
  - Mostly physics
- One of common tasks in game development
  - Source of errors and "glitches"

Video Evidence

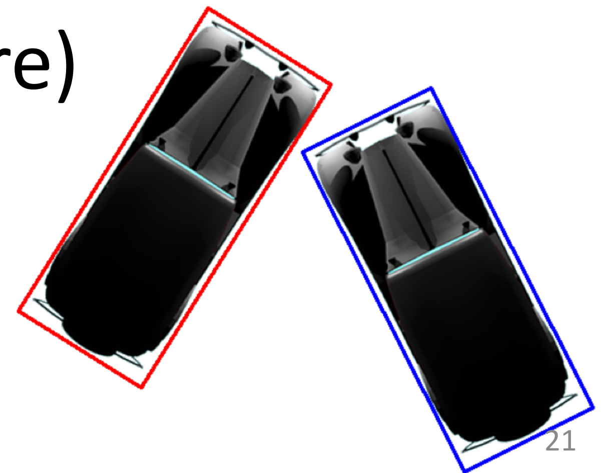
- Add a youtube video showing the error of collision
- https://www.youtube.com/watch?v=mYhNvOg5yJ0

Static vs Dynamic Objects

- Static objects don’t move; dynamic objects do
- Collision between a static and dynamic objects
  - Easier

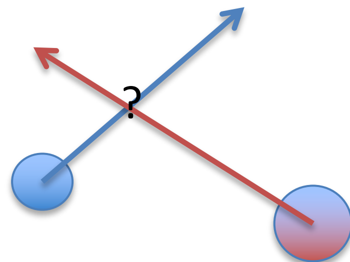


- Collision between two (or more) dynamic objects
  - Harder



Collision Detection: The Problem

- For moving objects
  - Did/will they collide? (bullet and target)
  - When did/will they collide? (cars)
  - First collision / all collisions (snooker balls / bricks)
  - Compute the collision *normal vector* (for response)
  - Depends on the game



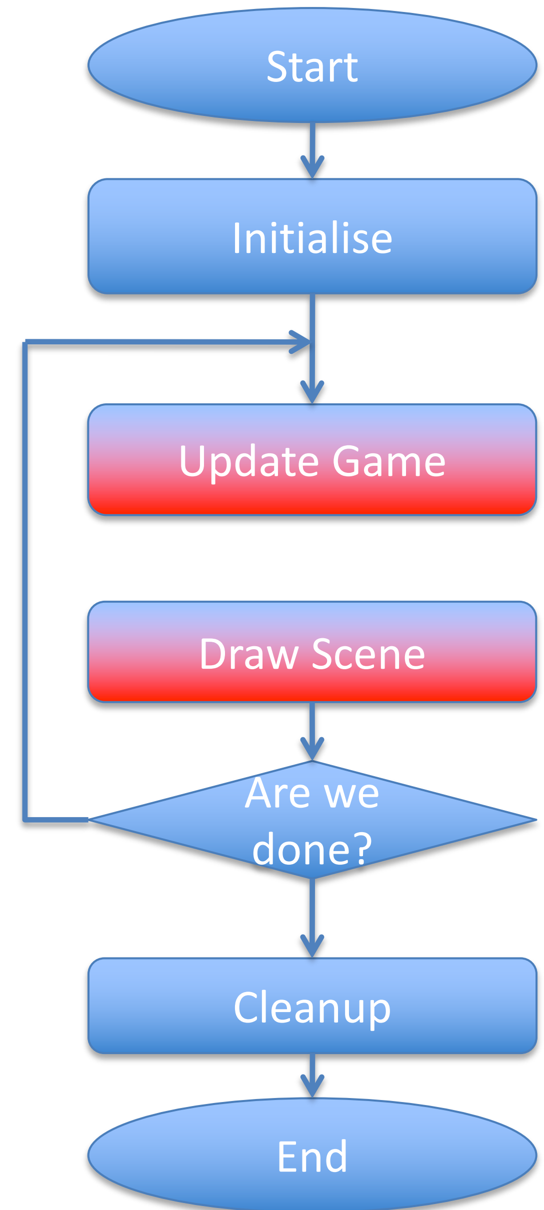
Given speed, shape, and time

Main Loop

Naïve approach:

```
for(i=0;i<num_obj-1;i++)
  for(j=i+1;j<num_obj;j++)
    if(collide(i,j)){
      react;
    }
```

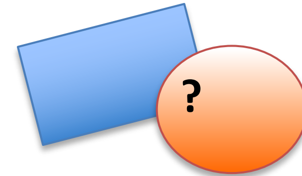
- Issues:
  - How
  - Can be **very** slow



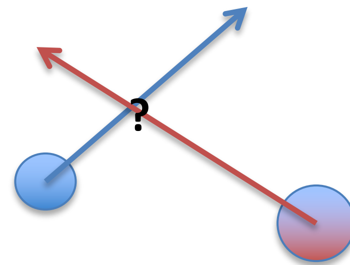
Collision Detection: How

Two basic techniques

- Overlap testing
  - Detecting whether a collision has already occurred
  - Most common technique

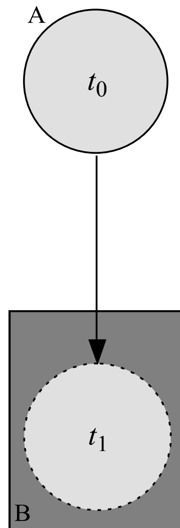


- Intersection testing
  - Predicting a collision



Overlap Testing: Collision Time

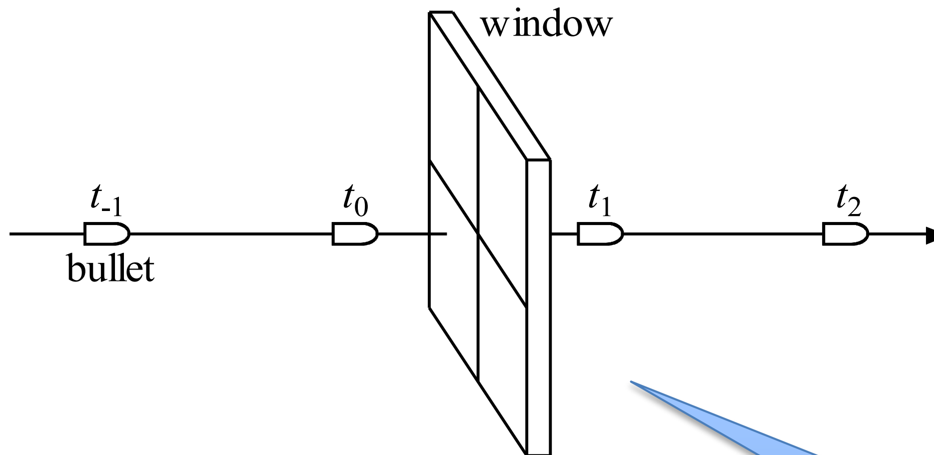
- Collision time can be calculated by moving object “back in time” until right before collision
  - Bisection is an effective technique



Initial Overlap Test

Limitations

- Fails with objects that move too fast
  - Unlikely to catch time slice during overlap



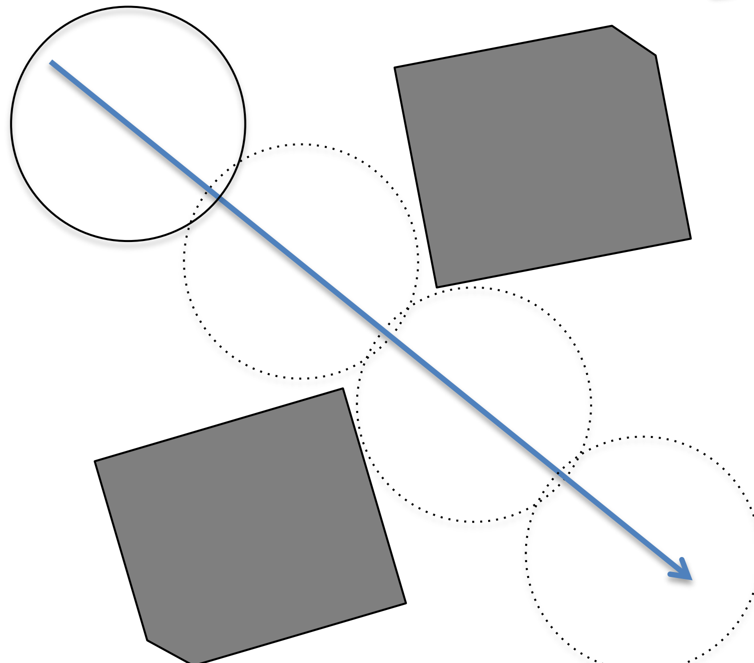
Leads to *interpenetration* and *tunnelling*

Glitches in Games

- Players/objects falling through
- Projectiles passing through targets
- Players getting where they should not get
- Players missing a trigger boundary

Caused by faults in collision detection

Hard to prevent due to the discrete motion



Possible Solutions

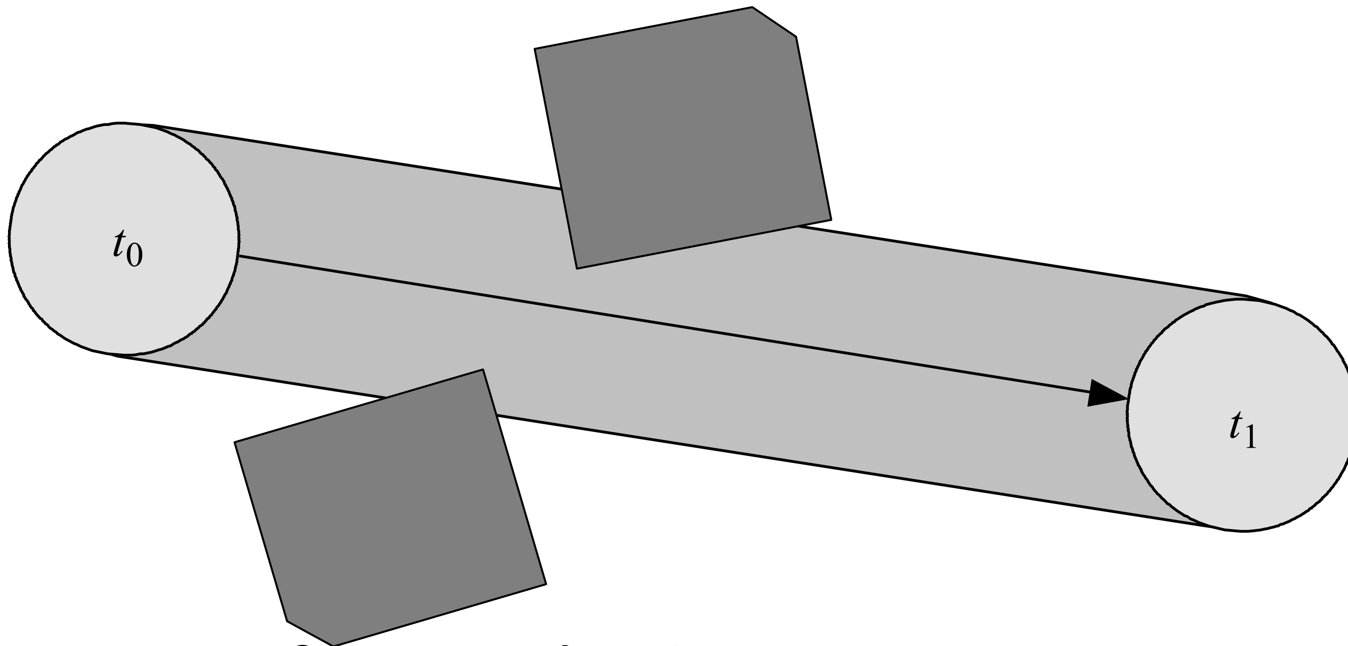
- Possible solutions:
  - Design constraint on speed of objects
    - May not always be feasible (bullets, etc.)
  - Reduce simulation step size
    - Hardware limitations, odd shapes
  - Intersection testing

Intersection Testing

- Predict future collisions
- When predicted:
  - Move simulation to time of collision
  - Resolve collision
  - Simulate remaining time step
- Assume constant speed (over some time)
  - Ideal for dynamic-static object collision

Example: Moving Sphere

- **Extrude** geometry in direction of movement
  – sphere turns into a "capsule" shape



- Then, test for overlap!

Limitations

- Issue with networked games
  - Future predictions rely on exact state of world at present time
  - Due to packet latency, current state not always coherent
- Assumes constant velocity and zero acceleration over simulation step
  - Has implications for physics model

Making It Work

- It is not feasible to test for every pair of entities if they collide
  - N^2 tests

- Therefore, usually we consider
  - Detailed view (colliding triangles and meshes)
  - Mid-level view (simplified geometry)
  - Global view (data structures to partition the entities)