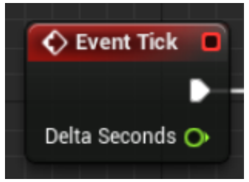
G54GAM Games

Building Games

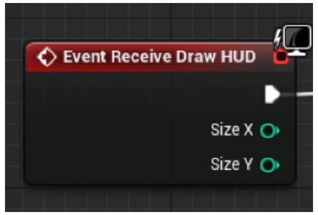
Physics

The Game Loop (attempt 2)

```
start game
start render thread
while( user doesn't exit )
{
    how much time has elapsed?
    get user input
    get network messages
    simulate game world(elapsed time)
    resolve collisions
    move objects
    play sounds
    sleep(desired-elapsed time)
}
exit
```



```
while( user doesn't exit )
{
    draw graphics
    sleep(desired-elapsed time)
}
exit
```

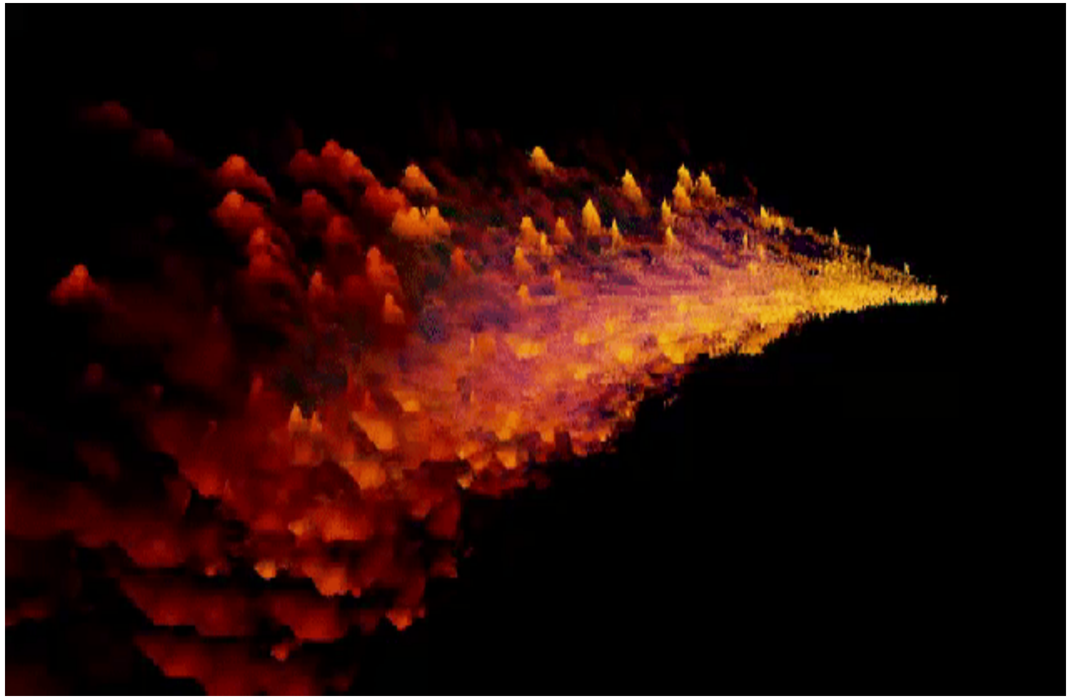


Physics Engine

- Responsibilities
  - Update and maintain positions and orientations of items
    - In response to user input
  - Determine what collisions are possible
  - Determine what collisions have happened
  - Generate data regarding those collisions
    - Drive game play events
  - Resolve those collisions
- *Physical* representations
  - Particle
  - Rigid Body
  - Soft Body

Particle Physics

- A particle based physics system doesn’t care about collisions, only motion
  - All objects in the system are particles
    - Dimensionless points in space.
    - They have no radius
  - Because they have no radius, they don’t collide
    - Might have nominal mass
      - Attract / repel one another
  - More applicable to graphical effects than physics behaviours

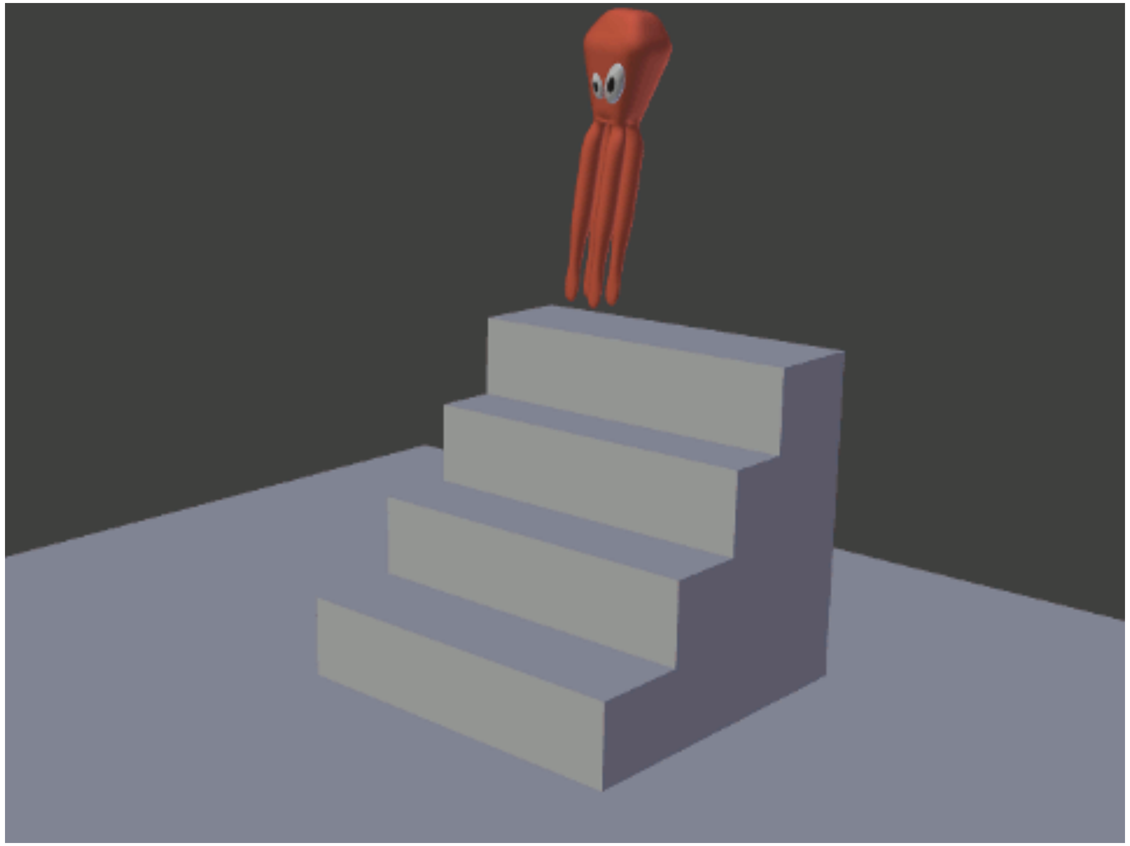


Rigid Body Physics

- Rigid bodies have a physical presence
  - A sphere, a cuboid, a height-map
- Defining characteristic is that they **don’t deform**
  - A balloon in the real world can be squeezed and stretched
  - A balloon represented in a rigid body physics system cannot
- Because they have physical presence, they have dimensionality
  - Radius, width, height,
    - A centre of motion
  - They can collide
  - They can can experience torque and rotate
- Computationally significantly less expensive
  - Than computing physical properties of soft bodies

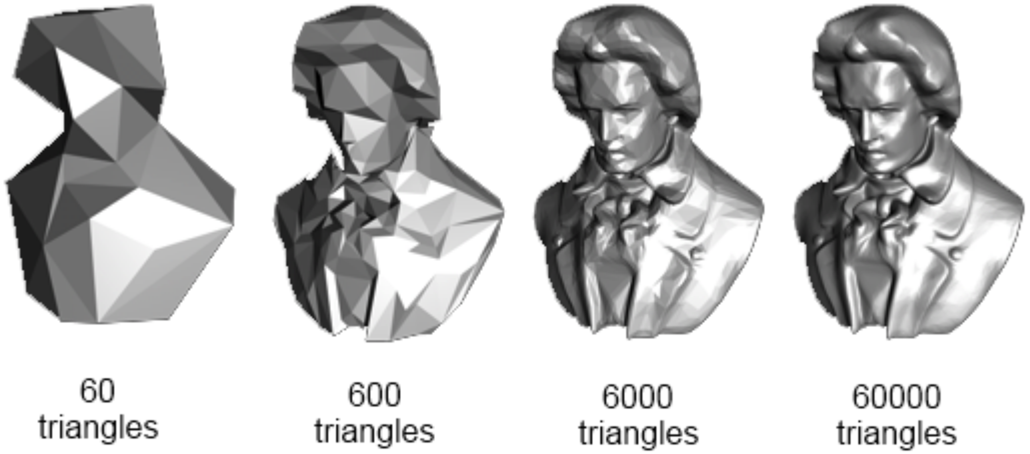
Soft Body Physics

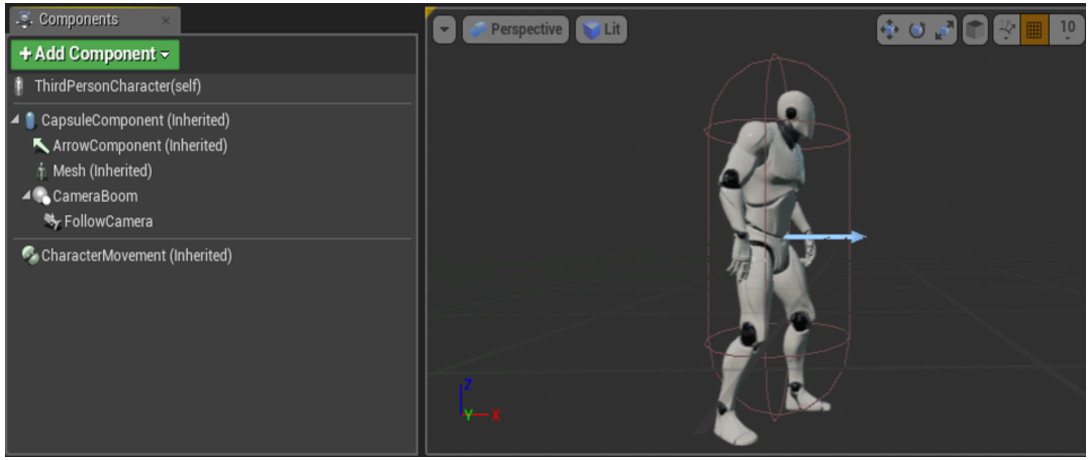
- Cloth, cushions, water
  - Anything that deforms when subject to a force
  - Most accurately represented as a soft body.
- Soft bodies must be discretised in a physics engine
  - Many interconnected small bodies
    - Connected via *constraints* (e.g. springs)
  - Significant computational expense
- Focused on situations where application enhances user experience/immersion



Physical Representation

- Physical representation of objects **does not need to precisely map to graphical representation**
  - Does need to be detailed enough for its interactions with the environment to appear believable
    - Capsule == character movement
  - Does not need to be as detailed as the graphical model
- Complexity of an object is a function of its number of components
  - Faces, primitives
  - We reach a point of diminishing returns in physics quickly



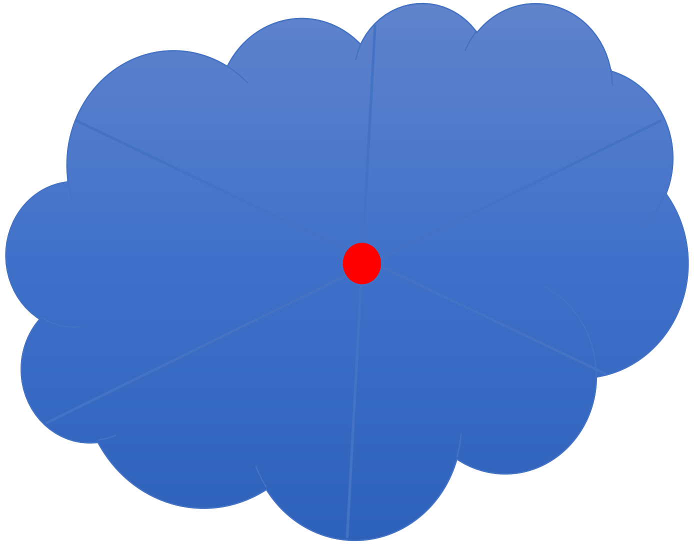

Components
+Add Component
ThirdPersonCharacter(self)
CapsuleComponent (Inherited)
ArrowComponent (Inherited)
Mesh (Inherited)
CameraBoom
FollowCamera
CharacterMovement (Inherited)
Perspective
Lit
10
Z
Y
X

Physics in Games

- “Simulate the game world”
    - Iterate the fundamental model
    - Moving objects based on **elapsed time**
        - Kinematics
            - Motion ignoring external forces
            - Only considering position, velocity
        - Dynamics
            - The effect of forces on the objects
- “Resolve Collisions”
    - Collisions between moving objects
        - Collision detection
            - Did a collision occur?
        - Collision resolution
            - Are we now in a restricted state due to constraints?
            - How do we fix it?

Physical Representation

- Typically ignore geometry
  - How the object is *physically shaped*
    - Also do not worry about how it looks
    - Only needed for collisions
  - Focus on how it moves
- Every object as a *point*
  - *Centroid*
    - The average of all points
    - Or the center of mass
    - Generally all objects have a uniform density
  - Often the default transform of the actor / entity



Kinematics

- Basic Motion
- Determine an object's displacement *s* at time *t*
  - Typically already know it from a previous time

$s'(t) = ds/dt = v$

- Assume constant velocity *v*

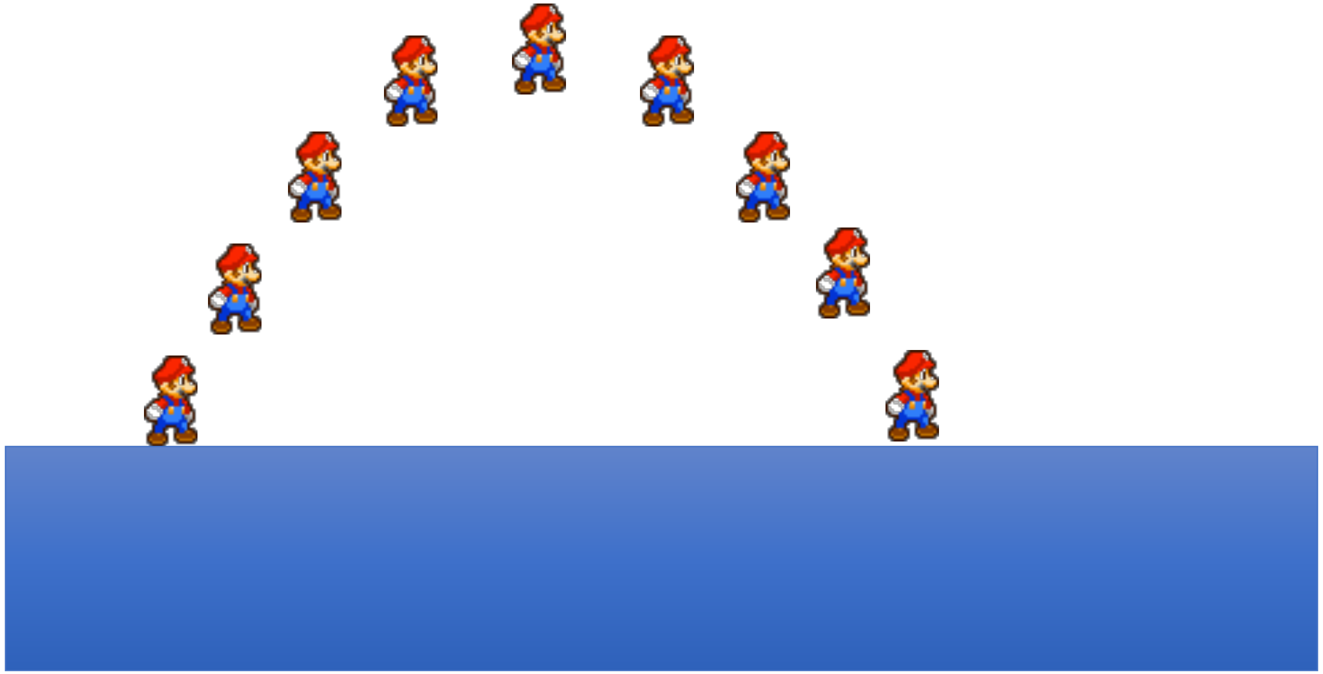
$s(t+\Delta t) = s(t) + v\Delta t$

$\Delta s = s(t+\Delta t) - s(t) = v\Delta t$

- What is *Δ* (delta) t?

Kinematics

- Acceleration
  - Rate of change of velocity over time
  - $a = dv/dt$
- Velocity
  - Rate of change of displacement over time
  - $v = ds/dt$
- Velocity is obtained
  - by integrating acceleration with respect to time
  - $v = \int a.dt$
- Displacement is obtained
  - by integrating velocity with respect to time
  - $s = \int v.dt$



Analytical Solutions

- Acceleration
  - rate of change of velocity
    - Rate of change of rate of change of position
- Integrate twice to calculate position

  $s(t)'' = g = -9.81\ m/s^2$

  $s(t)' = gt + v_0$

  $s(t) = 0.5gt^2 + v_0t + s_0$

  $s = ut + 0.5at^2$

- An analytical solution to solving an ordinary differential equation
  - A simple, closed-form function
  - Describes position for all possible values of time t
  - “Perfect” simulation of gravity / parabolic movement

Newtonian Linear Dynamics

- Newton’s laws of physics
  - A body will remain at rest or continue to move in a straight line at a constant speed unless acted upon by a force.
  - The acceleration of a body is proportional to the resultant force acting on the body, and is in the same direction as the resultant force
  - For every action there is an equal and opposite reaction
- Forces affect movement
  - Based on the mass m of an object
    - F=ma
    - Gravity, impulses, repulsion, inertia
  - *Constrained* by springs, joints, connections, other objects
  - Calculate changing velocity and acceleration from the forces applied over the entire frame
- Need a general solution
  - Generally cannot find closed-form solutions for movement under force for all values of time t
  - Force, acceleration are rarely constant
    - Function of position, velocity