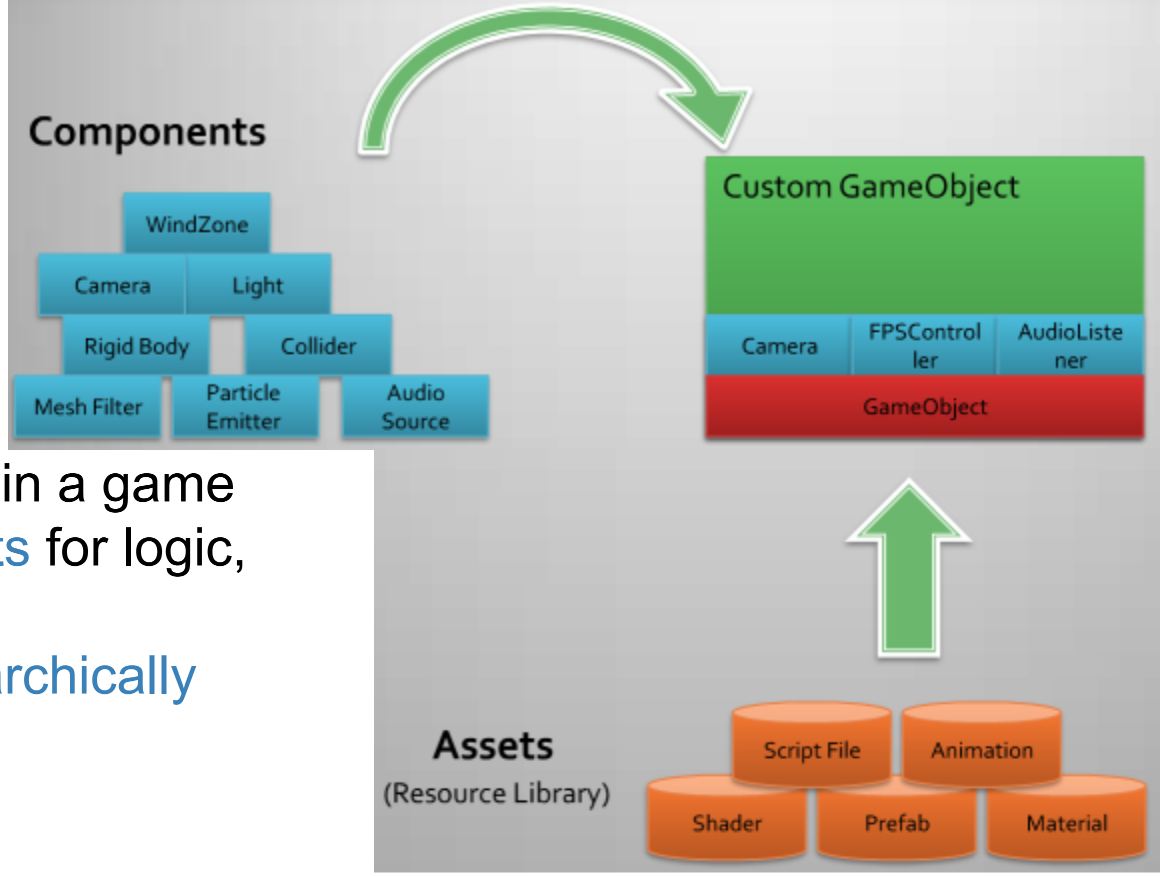
COMPSCI 715
Advanced Computer Graphics

Unity Scripting

Today's Mission

1. How does Unity scripting work?
2. How can scripts control GameObjects?
3. How do you apply this to your own project?

Recap: GameObjects



Games consist of them

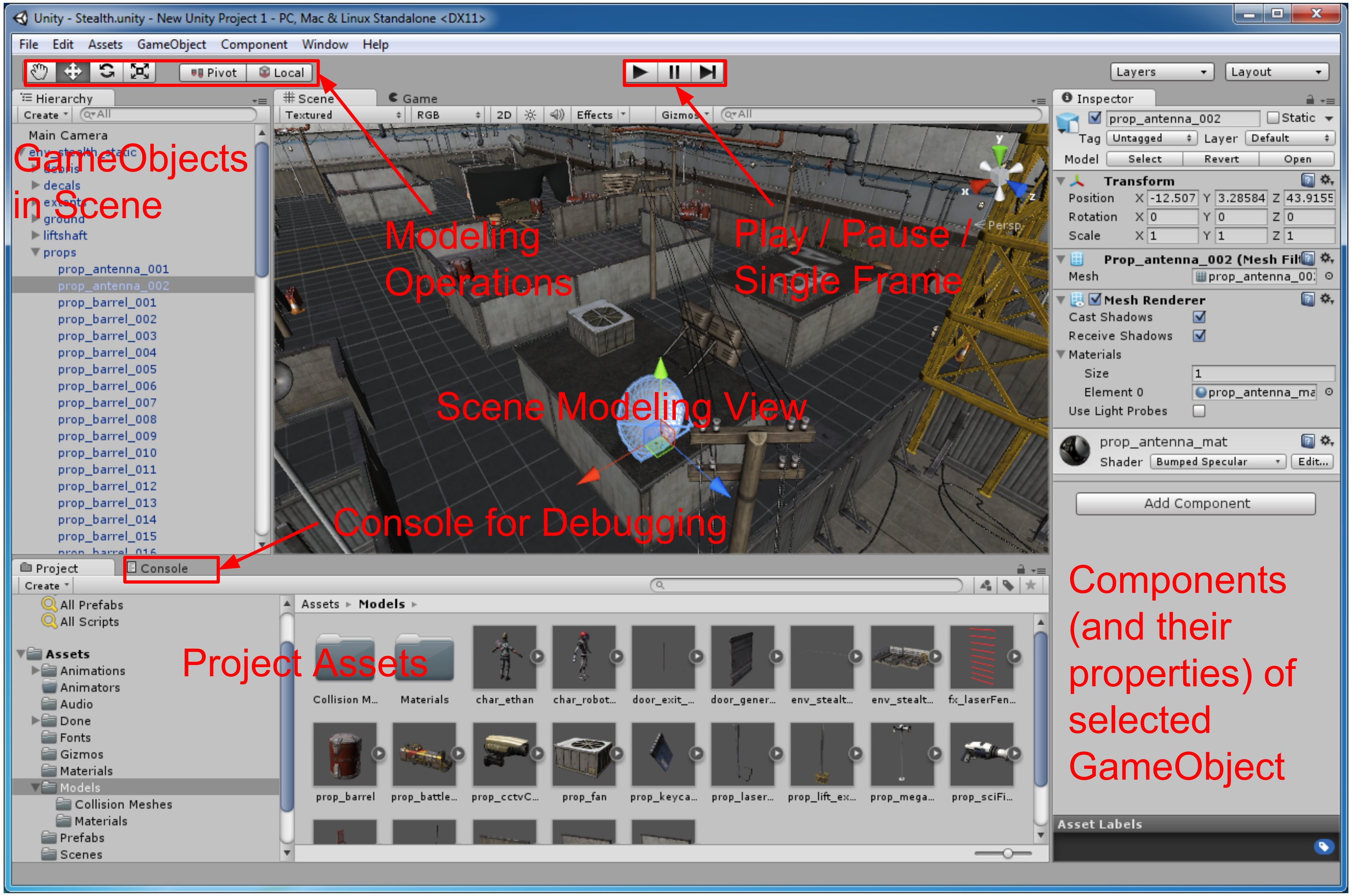
- Think of visible objects in a game
- But also invisible objects for logic, state etc.
- Can be organized hierarchically in a Scene

What a GameObject can do depends on its Components

- Technically Components are themselves objects
- Are just associated with GameObject and can reference it
- Give a GameObject more features by adding components, e.g. visual appearance, physics, dynamic behavior
- Knowing Unity's capabilities means knowing the different components

Thanks again to Michael Ivanov for his great figures: http://www.slideshare.net/sasmaster/unity3d-programming-5725801

Recap: Unity GUI


Unity - Stealth.unity - New Unity Project 1 - PC, Mac & Linux Standalone <DX11>
File Edit Assets GameObject Component Window Help
Pivot Local
Layers Layout
Hierarchy
Scene Game
Inspector
GameObjects in Scene
Modeling Operations
Play / Pause / Single Frame
Scene Modeling View
Console for Debugging
Project Assets
Components (and their properties) of selected GameObject
Project Console
Asset Labels
Add Component

Hierarchy: Main Camera, env_stealth_static, debris, decals, extents, ground, liftshaft, props, prop_antenna_001, prop_antenna_002, prop_barrel_001, prop_barrel_002, prop_barrel_003, prop_barrel_004, prop_barrel_005, prop_barrel_006, prop_barrel_007, prop_barrel_008, prop_barrel_009, prop_barrel_010, prop_barrel_011, prop_barrel_012, prop_barrel_013, prop_barrel_014, prop_barrel_015, prop_barrel_016

Scene: Textured, RGB, 2D, Effects, Gizmos, Persp

Inspector: prop_antenna_002, Static
Tag Untagged, Layer Default
Model Select Revert Open

Transform
Position X -12.507 Y 3.28584 Z 43.9155
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Prop_antenna_002 (Mesh Filter)
Mesh prop_antenna_002

Mesh Renderer
Cast Shadows ✓
Receive Shadows ✓
Materials
Size 1
Element 0 prop_antenna_mat
Use Light Probes

prop_antenna_mat
Shader Bumped Specular Edit...

Project: All Prefabs, All Scripts, Assets, Animations, Animators, Audio, Done, Fonts, Gizmos, Materials, Models, Collision Meshes, Materials, Prefabs, Scenes

Assets ▸ Models ▸
Collision M... Materials char_ethan char_robot... door_exit_... door_gener... env_stealt... env_stealt... fx_laserFen...
prop_barrel prop_battle... prop_cctvC... prop_fan prop_keyca... prop_laser... prop_lift_ex... prop_mega... prop_sciFi...

Script Components

Custom code

- Add Component -> New Script, select name and language
- Cogwheel -> Edit Script
- Choose name carefully (hard to rename)

Defines custom state & behavior

- State through object fields (properties)
    - Public properties visible in Inspector
- Behavior through event handler methods

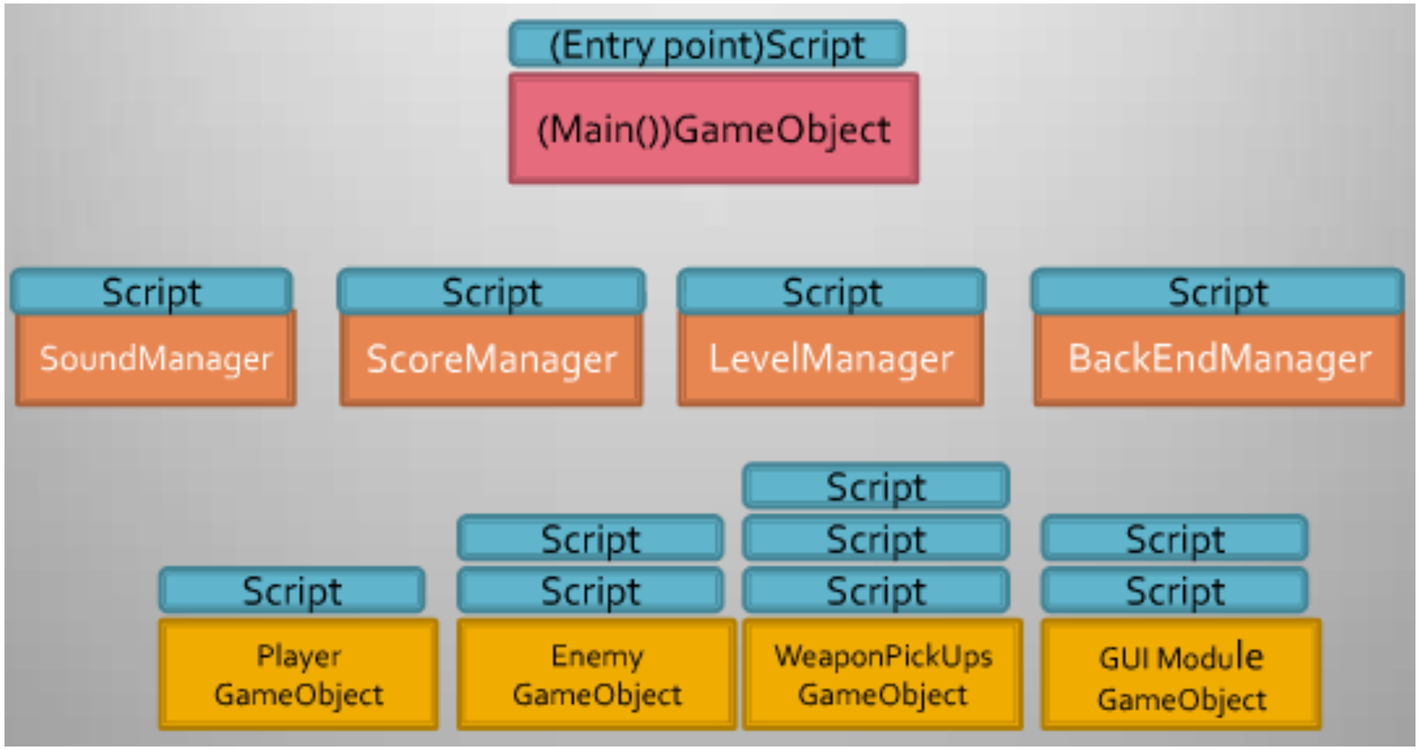
```
using UnityEngine;
using System.Collections;

public class MyScript : MonoBehaviour
{
   void Start () {
        // Used for initialization
   }

   void Update () {
        // Update game state here
        // Called once per frame
   }

   void OnCollisionEnter(Collision c) {
        // This GameObject has collided
        // with another object.
        // Do something!
   }
}
```

Typical Code Design



Note: non-visible GameObjects for managing game data and game state

Script Example: Collision turns on light

- Public property light can be set in Inspector
- gameObject gives the GameObject of a component
- Debug.Log() prints message onto console (see GUI tab)
- Don’t forget to build your script (F8 in MonoDevelop)

Coding Style:
Property names start lowercase and method names start uppercase

```
public class MyLightSwitch : MonoBehaviour {
    public Light light;


    void Start () {
        light.enabled = false;
    }


    void OnCollisionEnter(Collision c) {
        Debug.Log ("Collision with " +
                                c.gameObject.name);


        if (c.gameObject.name == "Player")
            light.enabled = true;
    }
}
```

Key/Button Input

Low-level: get key presses directly

bool down = Input.GetKeyDown(KeyCode.Space);

bool held = Input.GetKey(KeyCode.Space);

bool up = Input.GetKeyUp(KeyCode.Space);

Abstracted: let user specify control settings

Input Manager: Edit -> Project Settings -> Input

bool down = Input.GetButtonDown("Jump");

bool held = Input.GetButton("Jump");

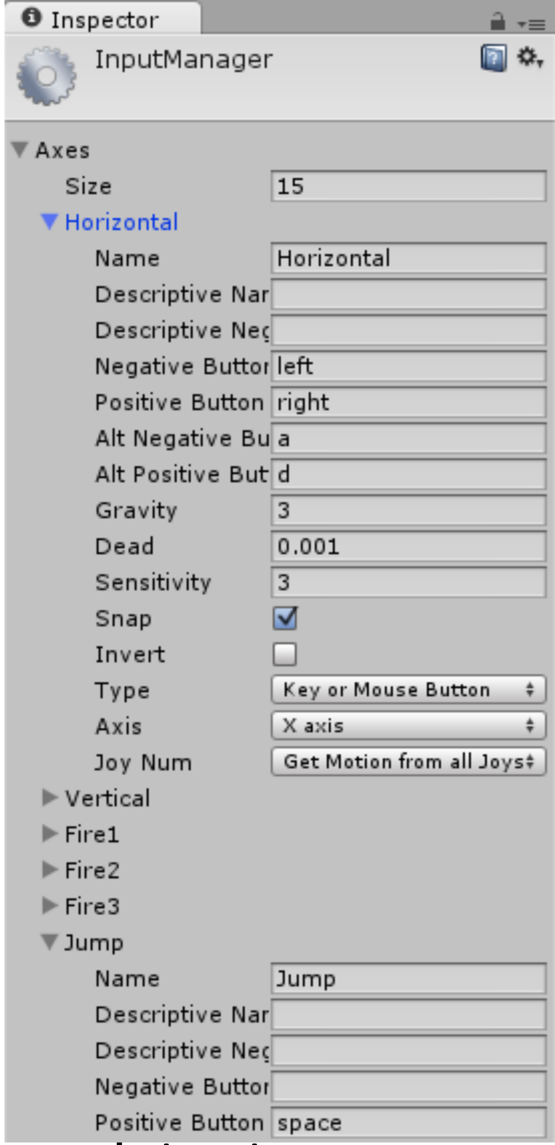
bool up = Input.GetButtonUp("Jump");

High-level: use configurable mechanics

Adjust behavior with properties: sensitivity, gravity and dead.

float h = Input.GetAxis("Horizontal");   // h in (-1, +1), same for Vertical

float h = Input.GetAxisRaw("Horizontal"); // either -1, 0, or +1 (discrete)



▼ Axes	
Size	15
▼ Horizontal	
Name	Horizontal
Descriptive Nar	
Descriptive Neç	
Negative Buttor	left
Positive Button	right
Alt Negative Bu	a
Alt Positive But	d
Gravity	3
Dead	0.001
Sensitivity	3
Snap	✓
Invert	☐
Type	Key or Mouse Button
Axis	X axis
Joy Num	Get Motion from all Joys
▶ Vertical	
▶ Fire1	
▶ Fire2	
▶ Fire3	
▼ Jump	
Name	Jump
Descriptive Nar	
Descriptive Neç	
Negative Buttor	
Positive Button	space

Script Example: Use Input to Control Transform

- Do translation and rotation for every frame in Update()
- Dependent on horizontal and vertical axis state
- Attenuate with speed settings
- Varying time between frames may cause jitter, so take actual time (in seconds) since last frame into account: Time. deltaTime

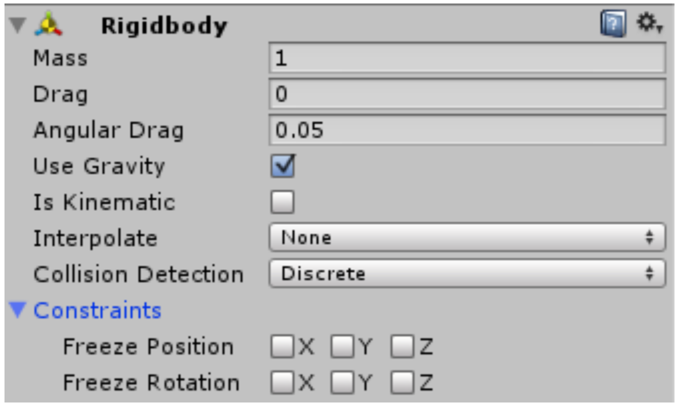
Note:
Transforms don’t mix with physics!
Use the physics engine to move a RigidBody.

```
public class PlayerControl : MonoBehaviour {
  float moveSpeed = 10f;
  float turnSpeed = 50f;

  void Update () {
    transform.Translate(  // uses local axes
      Input.GetAxis("Vertical")
     * Vector3.forward  // = (0, 0, 1)
     * moveSpeed     // in m/s (not m/frame)
     * Time.deltaTime);

    transform.Rotate(  // uses local axes
     Vector3.up,  // = (0, 1, 0)
     Input.GetAxis("Horizontal")
     * turnSpeed * Time.deltaTime);
  }
}
```

Scripting Physics



Rigidbody	
Mass	1
Drag	0
Angular Drag	0.05
Use Gravity	✓
Is Kinematic	☐
Interpolate	None
Collision Detection	Discrete
Constraints	
Freeze Position	☐X ☐Y ☐Z
Freeze Rotation	☐X ☐Y ☐Z

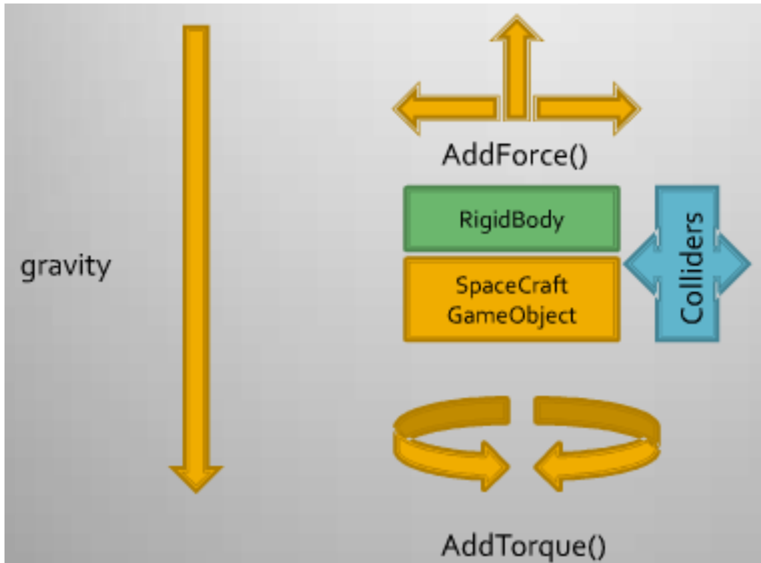
Always change physics in FixedUpdate()
as this is synced with physics engine!

RigidBody properties in Inspector plus others:
velocity (units or M per sec), angularVelocity, centerOfMass, …

RigidBody methods:

- Usually move object indirectly by applying force to it, e.g. AddForce(x, y, z) or AddRelativeForce(lx, ly, ly)
- Change velocity only directly for immediate change:

```
if (Input.GetButtonDown ("Jump")) {
  rigidbody.velocity = Vector3(0,10,0);
}
```



Script Example: Use Input to Control Physics

- GetComponent<ComponentType> () to get a certain component of this GameObject
    - Slow, so better not in (Fixed)Update
- Add force in FixedUpdate (!)
    - Force causes acceleration
    - Experiment with values to find right one
- Limit speed by checking & truncating length of velocity
    - But this will also limit effect of gravity, explosions etc.

```
public class PlayerControl : MonoBehaviour {
  float accel = 5f;    // acceleration
  float maxV = 2f;     // maximum speed (m/s)
  Rigidbody rigidBody;  // body to move
  void Start() {
    rigidBody = GetComponent<Rigidbody> ();
  }
  void FixedUpdate () {
   rigidBody.AddForce (
      Input.GetAxis("Horizontal") * accel,
      0,
      Input.GetAxis("Vertical") * accel);
   if(rigidBody.velocity.magnitude > maxV)
     rigidBody.velocity =
      rigidbody.velocity.normalized * maxV;
  }
}
```