What this section covers…

- Using 3D with 2D controls (2.5D game).
- Using world-space UI in a 3D game.
- **CrossPlatformInputManager** & virtual controls.
- Saving game state to PlayerPrefs.
- UI anchors + much more.

In This Video...

- Why you may want to use version control.
- An overview of how we use it.
- Follow us on GitHub.
- Where to find the course repositories.

Version Control Glossary

- **Repo**: Short for repository. The code for a project.
- **Commit:** Save local snapshot of your project.
- **Push:** Send your local repository to the server.
- **Pull**: Get your remote repository from the server.
- **Checkout:** Load local snapshot of your project.

Follow Us On GitHub

- Signup for GitHub if you haven't already.
- Visit https://github.com/CompleteUnityDeveloper
- Take a look around the site.
- Click through to Ben / Brice.
- Follow us for future code updates.

Sharing Your Game With Git

@UnityCourse
facebook.com/UnityCourse

In This Video...

- How Git can help you share your project.
- How we use Git for the course.
- What's different about how you may use it.
- Read Dan's blog post*

http://leereilly.net/2012/11/29/hosting-games-on-github.html

Open Our Project Prototype

- Visit https://github.com/CompleteUnityDeveloper
- Find the 10-TwinStick “repo”.
- Download the latest commit from the **pt** branch.
- Open the project in Unity to test it works.

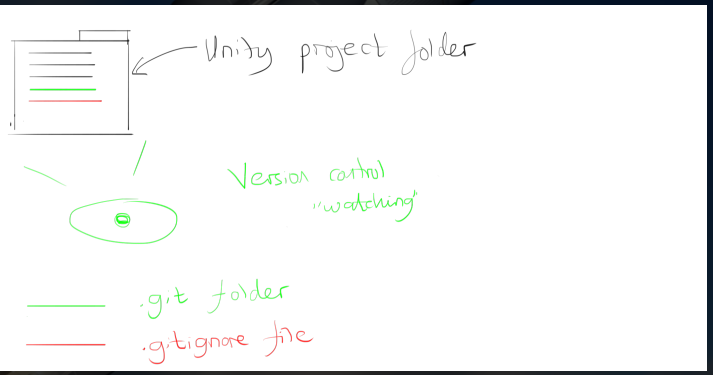
In This Video...

- Download SourceTree
- Creating local & remote repos.
- Using a **.gitignore** file for Unity.
- Connect to GitHub (or BitBucket).
- Share your repo in the discussions.

Bitbucket vs GitHub

GitHub	BitBucket
+ Very well known, great support on web.	- Less well known.
+ Plays well with SourceTree.	- Can have problems with SourceTree!
- Private repos are paid.	+ Private repos are free.

Ecosystem Overview



Share Your Repo

- Put a secret message in your scene.
- Push your repo to GitHub (or SourceTree).
- Share it in the Discussions.
- Challenge people to find the message.
- Celebrate, you're now a real coder!

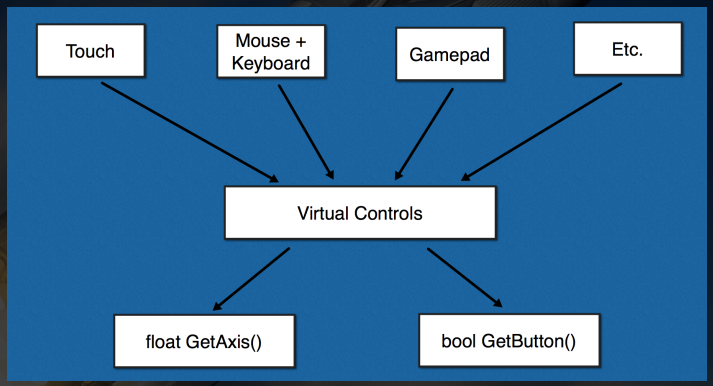
Using CrossPlatformInputManager

@UnityCourse
facebook.com/UnityCourse

In This Video...

- What is **CrossPlatformInputManager**.
- How a virtual control layer works.
- Setting-up and reading control values.

Virtual Control Layer



Using CrossPlatformInputManager

1. Assets > Import Package > CrossPlatformInput
2. using UnityStandardAssets.CrossPlatformInput;
3. Use **CrossPlatformInputManager.** to access.

Log Virtual Control Values

- Lookup how you read input using **Input.**
- Replace with **CrossPlatformInput.**
- Import the appropriate namespace.
- Print control values to the console.

Using Analog Gamepad (Optional)

In This Video...

- About using gamepad input.
- Setting up a PS4 controller on a Mac.
- Using an Xbox controller on a PC.
- Using the input gravity setting.
- Other input settings such as sensitivity.

Using A Gamepad OR Gravity

- If you have a gamepad, try and get it working.
- Otherwise simulate using input gravity.

In This Video...

- Import the Characters standard asset pack.
- Use the RollerBall prefab game object.
- Using physics freeze position constraints.
- Explore the control options.

Explore The RollerBall

- Explore the RollerBall for your 2.5D game.
- Try all the control settings.
- Look at the scripts, see if you can modify them.
- Ask any questions in the discussions.

In This Video...

- An overview of replay systems.
- Deterministic vs. non-deterministic replays.
- An overview of how we will do things.
- Introducing a circular (ring) buffer.

Deterministic Event Storage Replay

- Store all starting state (seeds, AI, etc etc).
- Reproduce all player input.
- Less storage, but simulation must run perfectly.
- Hard to do a rewind.

Non-Deterministic State Storage

- Store entire game state every frame.
- Larger file sizes, but allows random access.
- Allows rewind easily too.

Research replay solutions

- Search for “saving replay Unity” or similar.
- Write-down the outline of a solution.
- Continue the video, and I’ll introduce my idea.

In This Video...

- Why we’re not using Unity’s **Keyframe** struct*
- Comparing **class**es and **struct**s as options.
- Creating our own **MyKeyframe** struct.

http://docs.unity3d.com/ScriptReference/Keyframe.html

Write A MyKeyFrame Class

- In the **Replay.cs** as a helper class.
- Define as a class (not struct) for now.
- Try and provide a “constructor”, so you can say **keyFrame = new MyKeyFrame (time, pos, rot)** elsewhere in your code. Bonus marks!

In This Video...

- An overview of our replay system code.
- Implementing a ring buffer for frames.
- Testing our record.

In This Video...

- Create a simple Game Manager.
- Use it to keep track of recording / playback.
- Wire it to the **ReplaySystem.cs** script.
- Test playback.

Write GameManager.cs

- Create an empty game object.
- Attach **GameManager.cs**
- Have it keep track of **bool recording**.
- While holding “Fire1” button, is in playback.
- Otherwise in record mode (normal gameplay).

Make ReplaySystem.cs Read Mode

- Get your **ReplaySystem.cs** reading state.
- Ensure “Fire1” goes into playback (Ctrl key).
- Celebrate!

In This Video...

- Using the **MobileSingleStickControl** prefab.
- Creating a button to trigger the replay.
- A little more about UI anchors.
- Building to Android & testing performance.

Setup A Replay Button

- Make a replay button.
- Choose a relevant sprite (may need to rotate).
- Test this button triggers the replay.

Level Unlocks In Unity

@UnityCourse
facebook.com/UnityCourse

In This Video…

- Dig-up your old **PlayerPrefsManager**
- Add code to handle level unlocks.
- Create a simple proof of concept.

Demonstrate Level Unlocks

- Convince yourself the system works.

In This Video...

- See one way of pausing your game in Unity.
- Look at **OnApplicationPause** message.
- Understand **Time.timeScale** better.

Add A Pause System To A Game

- Pick a past project.
- Add pause functionality.
- Persist until it is done.


Section Wrap Up