Charles Sturt University

IT Masters
itmasters.edu.au

Unleash your inner Game Dev
ITE529 Free Online Course

Week 1

Charles Sturt University

IT Masters
itmasters.edu.au

ITE529: Cross Platform Game Development

This short introduction course teaches students the basic principles of mobile game development using Unity, a powerful cross-platform game engine that utilises C# for high-level game logic, creating smooth pipelines between desktop, console and mobile devices.

Unity is the engine behind titles like Infinity Blade, Angry Birds 2, Crossy Road and many more.

Charles Sturt University

IT Masters
itmasters.edu.au

Mentor

Jamie T. Bentley

- Visual Effects Artist & Technical Director of Post Production at Method Studios (formerly MRPPP)
- Worked on over 50 commercials for Holden, Mazda, Ford, Cadbury, Mars, Australian Defense Force, Australia Post, etc.
- Senior Software Engineer @ Playside Studios
- Previously Technical Director of The Binary Mill
- Over 20 Mobile games released, also PC & Arcade
- Over 40,000,000 downloads
- Titles include Spongebob - Sponge on the Run, Mini Motor Racing, Fishy Bits, Gun Club and many more.

Unity Workflows

Unity projects are usually built upon the following basic elements

- Art
- Game Logic
- Audio
- User Interface & Input
- Metrics & Monetisation

Art

Art can be broken into smaller subcategories, all of which are handled differently within Unity.

3D Art
Usually brought in from *3D Max, Maya, Softimage XSI* or *Blender*. This is typically the backbone of any non-2D game and used for everything from level design to characters and everything inbetween.

3D objects can be animated from within their original source, or animated directly within Unity. Unity also supports complex character rigs.

Art

2D Art
For User Interface, particle effects and sprites. Although the final rendering method for 2D art is typically a quad rendered in 3D, the process is still referred to as two dimensional.

Assets will typically come from *Photoshop*. Unity supports numerous file formats, and while *PSD* is the easiest to use for artists because it keeps layers intact at the cost of larger file size in the project.

Unity compresses textures into native gpu-supported formats (*DXT, PVRTC, ETC*), so it is best to keep your art uncompressed at all times. *PNG* is quite efficient at this task and also supports alpha channels.

Audio

Audio is typically broken up into user interface, sound effects and music categories. Unity supports many file formats including older tracker formats such as *Scream Tracker*, *FastTracker*, etc.

Typically, short sound effects should be decompressed in RAM to improve playback performance. Longer audio streams should be compressed & streamed from disk dynamically to avoid excessive memory usage.

Unity supports 3D Audio positioning, and will position a mono sound in space (depending on the amount of speakers), but a stereo sound will always be played as-is with no directional adjustments.

Game Logic

Game logic is the basis of any Unity project. With the exception of some basics, anything that responds to input or conditions is controlled by code.

Unity supports *C#, UnityScript* (a fork of *Javascript*) and *Boo* scripting languages.

Due to most third party SDKs utilising .NET, C# is the language of choice for industry professionals. Javascript, Boo and C# have communication issues with each other due to compilation order. This problem is fixed by using a unified C# pipeline.

Game Logic

Game Logic for video games can be broken up into distinct categories

- Player Movement, Interaction and Input
- Enemy Behavior
- World Behavior
- Meta-Game Behavior
- File Handling
- Third Party Integration (Steam, FMOD, In-App Purchasing, Google Play, Ads)

Game Logic

Player Movement, World Interaction & Reaction

Regardless of the game genre, input from the player must interact with an actor or an interface. This could be a character in a first person shooter or an entire battle field in a Real Time Strategy game.

Every game will use a unique design to meet these challenges. Rarely will a template script suffice.

The player's character in a game may have complex interactions with it's environment, and the functionality of this is the most likely to change during the production of a game.

Game Logic

File Handling

Games often require some relatively simple file I/O operations. These can be very straightforward like a savegame, or more complex and requiring synchronisation via network.

Common file handling tasks

- Loading/Saving Game State
- Sharing Data Between Devices
- Storing & Loading Large Data Sets (Inventories, Player History)
- Streaming in Game Assets (Music, Textures, etc)
- Dynamic Loading of Text Assets (Language Localisation)

Game Logic

Third Party Integration

Games invariably require some level of participation with third party code. In the console world, this might simply be supporting Xbox Live or PSN features. For PC it will most likely be for technology partners (Havok Physics, SpeedTree) or distribution partner feature integration (Steam, Origin, etc).

Game Logic

Any mobile game may have any of the following third party SDKs implemented;

- Ad Serving (Unity Ads, Admob, AdColony, etc)
- Analytics (Unity Analytics, DeltaDNA, etc)
- Networking (Unity Networking, Photon Networking, Photon Bolt, etc)
- Scoreboards & Social Media (Facebook, Twitter, Gamecenter, Google Play Services, Instagram, etc)
- Many, many more.

Game Production

Game Production

Most game productions follow a similar path, although occasionally some games will have more pre-production planning than others. Games that borrow many gameplay mechanics from other titles ('clones') often require less planning.

Game Production

At it's highest level, game production works through these basic steps.

1. Concepting & Gameplay Goals
2. Proof of concept prototype (with limited art assets)
3. Development of Game Logic & Art
4. Revise & Improve Gameplay
5. Revise & Improve Art
6. Return to Step 4 (if time/money allows)
7. Quality Assurance & Optimisation Pass
8. Release!

Every company and every game has it's own pipeline, this is only a guide.

Next up...

Unity interface overview...