Week 9: Game Engines and Assets

DIGITAL ASSET DEVELOPMENT

Contents

- Game engines
- Game engine components
- Unreal Engine concepts

Game Engines

- For the remainder of the module, we will focus on assets for games
  - Games are a major market segment
  - All the asset types covered previously are relevant for game creation
- To be used within a game, assets must be imported into a game engine
  - The engine is used to assemble a game from (mostly) pre-prepared assets

Why Use an Engine?

- Modern games are highly complex, typically involving hundreds of assets
  - Performance of physics and lighting calculations is crucial
  - High quality graphical output is required
- Rewriting the tools to achieve this for every game would be hugely wasteful
  - Game engines allow reuse of algorithms and core functionality across the industry

Commonly Used Engines

- There is a vast array of engines in use for game production today
  - Use of many is restricted to major studios
- Some include a freely available level editing environment
  - Some of these also include game production functionality
- Of the latter, the best known are Unreal Engine and Unity

Engine Components

- A game engine consists of a number of functional elements:
  - A render engine to convert game data into graphical output
  - A physics engine to allow realistic motion
  - Basic game logic, such as input handling, collision detection and game AI
  - Scripting capability
  - An audio engine to handle sound output

Rendering

- A major issue for game engines is that rendering has to be done in real time
- To allow this, many short cuts are used
  - Hardware accelerated graphics take some calculations away from the CPU
  - Many advanced rendering functions (such as global illumination) are “faked”
  - Lighting can be baked into a level, avoiding the need for repeated calculations

Game Physics

- The physics engine allows the dynamics of game objects to be simulated
- Various types of calculation involved:
  - Rigid body dynamics (eg. falling bricks)
  - Soft-body dynamics (object deformation)
  - Fluid dynamics (eg. moving water)
- More realistic behaviour will take longer to simulate
  - Again, short cuts are frequently used

Game Logic

- The key attribute of a game engine is that it creates a world with rules
  - Without the rules to enable gameplay, an engine is really just a level editor
- Creating environments with inbuilt game logic gives a huge time saving
- Typically, templates are provided for different game genres
  - First person shooter, driving, platformer,...

Scripting

- All game engines require some form of coding to be available
  - Without this, new game functionality cannot be created
  - Also allows 3rd party plugins to be built
- Usually involve an OOP language such as C++, Java or C#
- Some also offer more visual tools for scripting game activity

Unity

- Unity is a powerful and flexible engine
  - Cross platform, with greater emphasis than most on mobile and web games
  - 2D and 3D development modes
  - Uses C# and UnityScript for scripting
- Personal edition is freely available to home users
  - Also to businesses with turnover less than $100000
  - Larger companies must buy Unity Pro

Unreal Engine

- Originally developed in 1998 for the first-person shooter game Unreal
  - Used for production of other FPS games
  - Released as an editor for players to create their own levels or mods
  - Licensed as a full game engine in 2009 under the name UDK
  - Free to download and use
  - Under licence agreement, 5% of profits made go to Epic Games (the developers)

Unreal Technology

- Still has an FPS bias, but can produce a wide range of games and apps
- Very effective lighting and rendering capability
  - Arch-vis is becoming a major area of use
- Uses C++ as a scripting method (the engine is built in C++)
- Blueprint system used for visual scripting (replaced UnrealScript)

Core Unreal Concepts

- Project: the files and folders making up the game, referenced in a project file
  - .uproject file format
- Level: a defined space within the game
- World: collection of all game levels
- Actor: any item that can be placed in a level; all actors have a transform matrix
- Component: a piece of functionality that can be applied to a specific actor

Unreal Game Concepts

- Pawn: any in-game avatar
- Character: a pawn that is playable
  - Incorporates player control setup, collision detection and bipedal movement
- PlayerController: links player input to game activity (one instance per player)
  - AIController: PlayerController for NPCs
- GameMode: the rules of the game; only one GameMode can exist per level

Actor Types

- StaticMeshActor: a piece of geometry of constant shape
- SkeletalMeshActor: geometry with a skeleton that can be animated
  - Commonly used for characters
  - Also machinery or deformable objects
- Lights: point, spot and directional
- Trigger: causes an event to occur in response to some action (eg. collision)

Asset Creation

- Some asset types are usually created within Unreal
  - Levels, materials, lights, particle systems
- Some are usually imported:
  - Meshes (static and skeletal)
  - Skeletal animations (eg. run or walk cycles and other character moves)
  - Textures
  - Sound files

Learning Unreal

- We will not be creating Unreal games in this module
- Instead you will be:
  - Learning the interface and workflow
  - Exploring tools to create and edit basic assets in the engine
  - Importing assets of various types
  - Creating simple levels and adding basic functionality