High-End 3D Visualization with CityEngine, Unity and Unreal

Pascal Mueller
Agenda

- Introduction into Game Engines (including Unity-Unreal Comparison)
- User Examples
- GIS2VR workflow: CityEngine to Unity
- GIS2VR workflow: CityEngine to Unreal
- Unreal Basics & Principles
- CityEngine VR Template
Introduction into Game Engines

Unity vs Unreal
Architectural Visualization

“Unreal Engine is 4th most used production renderer (including offline renderers)” CGarchitect 2018 Survey

Industry is using game engines for high-end 3D real-time visualization
What is a Game Engine?

A SW dev environment designed for people to build video games:
- Rendering engine for 2D or 3D graphics
- Physics engine, collision detection
- Spatial audio rendering
- Animation, artificial intelligence
- Scripting, scene graph
- Video support for cinematics
- Networking, memory mgmt, threading, localization support …
Why Game Engines?

A SW dev environment designed for people to build video games
An environment designed for people to build experiences

Experience != line of sight tool where you can do multiple things with
Experience = use-case specific intuitive real-time UX e.g. *find sniper*

…*BUT: game engines don’t support geo-referencing!!!*
Unity vs Unreal

Both are for free as long as you basically don’t publish “exe”
Unity vs Unreal: Tech

- Unreal more built-in features (Blueprints, Material Editor, etc.)
  - Unity catching up on features (Material Editor)
  - Some available via plugins (not all of them for free though)
  - Replicating high-end features of Unreal can be complicated in Unity

- Unreal default materials and assets are of higher quality

- Unreal tends to better suited for HQ/performance applications
  - E.g. global illumination system of Unity not dynamic and slower
Unity vs Unreal: Development

- Unreal has built-in visual scripting called Blueprint
- Development in Unity in C# & JavaScript and in Unreal in C++
- Source code for Unreal is public for everybody!!!
- Unreal exceptionally good for devops & source code mgmt.
- Unity supports more platforms (Nintendo Switch etc)
- Both support browser - but stability & performance lacking…
Unity vs Unreal: Adoption

• Unity tends to be more beginner friendly
• Unity has more indie developers i.e. asset store of Unity is bigger
• Epic Games supporting non-game applications more
• Unreal about 3 times more used than Unity in architectural viz
CityEngine User Examples
Vrbn, City of Zurich, HOK
GIS2VR: CityEngine to Unity

FBX-based Workflow
CityEngine: Select, «Export Models...»
Choose FBX, Tweak Settings, & Click Finish
Unity: Drag & Drop FBX to Assets & into scene
Move «Main Camera» to center
Enable VR

- Edit, Project Settings, Player
  - Check «Virtual Reality Supported»
- Add OpenVR Support
  - Click «+», «OpenVR»
  - Drag above «Oculus»
Start in VR: Press «Play» and experience it in HTC Vive
VR: Add Locomotion

- Window, «Asset Store»
- Search «SteamVR Plugin»
- «Import»
- «SteamVR_Settings» Dialog
  - «Accept All»
Add «Player»

- Delete «Main Camera»
- Drag in «SteamVr\InteractionSystem\Core\Prefabs\Player»
Add «teleport» prefab

- Drag in «SteamVR\InteractionSystem\Teleport\Prefabs\Teleporting»
Add Teleport Points

- Drag in multiple
  «SteamVR\ InteractionSystem\ Teleport\ Prefabs\TeleportPoint»

- Optional: Increase their scale
In VR, Teleport Using Click on Wheel
Teleport anywhere

- Click Mesh in Assets
- «Generate Colliders»
- «Apply»
Teleport anywhere

- Double-click street
- Copy/Paste
- Drag
- «SteamVR\InteractionSystem\Teleport\Scripts\TeleportArea»
  on street copy
In VR, Teleport Anywhere on Street
GIS2VR: CityEngine to Unreal
Datasmith-based Workflow
Unreal Material Handling in CGA Rules

In CityEngine
• (Optional) Modify CGA rules to assign appropriate Unreal materials
• Export scene with Datasmith exporter

In Unreal
• Create CityEngine Template project
• Import Datasmith file into Unreal
CityEngine to Unreal Workflow Overview

- `set(material.shader, «path/to/material/in/unreal/»)` in CGA

```python
set(material.shader, "/Game/Materials/CityEngineMaterials/M_CE_MaskedOpacity")
```

- **Template has 3 uber-shaders** (opaque, semi-transparent, masked)
- Or create custom materials in Unreal and reference them from CGA
Exporting from CityEngine

- Options to modify mesh granularity
- Choose between standard instancing & foliage system
- Datasmith takes care of game engine optimizations
Importing into Unreal

- Create CityEngine Template Project
- Open project and import Datasmith file
- Default options fine for normal sized meshes
- For lightmap baking the lightmap resolution might need tweaking
Unreal Basics & Principles

Quickstart Guide
Material Handling

- De-facto standard for material definition
- Graph-based editor UX
- Custom nodes can be created with HLSL code

adding glow to M_CE_Opaque
Visual Effects: Height Fog
Visual Effects: Atmospheric Fog
Postprocessing

- Depth of Field
- Lens effects
  - Vignette
  - Chromatic Aberration
  - Bloom
- Color Grading
- Ambient occlusion
  ...
Development Example: UI Popup

- Popup with metadata (CSV) exported from CityEngine
Widget Designer (UMG)

- Can either be HUD based or in-game UI elements
- Functionality can be added using Blueprints
Blueprints or C++?

Typically computational expensive methods should be written in C++, everything else via Blueprints

- Create C++ Actor class with special annotations
- Create Blueprint Actor based on this C++ class
- Call function from Blueprint
```cpp
UCLASS()
class BLUEPRINTCPPTEST_API AMyActor : public AActor
{
    GENERATED_BODY()

    UFUNCTION(BlueprintCallable)
    void ExpensiveFunction();
}
```
CityEngine VR Template

Work in Progress
Based on the Tabletop UX

- Common UX pattern in MR, VR & AR
- Collaborative
- Intuitive, people relate to table
- Specific for selected urban planning workflows
Using the CityEngine VR Template

- Create CityEngine VR Template Project
- Open project and import Datasmith file (optional: import own room)
- Downscale city to fit in Room