

Unreal Engine 4 Introduction



Agenda

- Unreal Engine evolution
 - Project setup
 - Game logic creation tools
 - UI creation tools
 - Automation
 - Physically based materials
- Mobile development with UE4
 - Performance tiers
 - General performance guidelines
 - Content creation notes

Project Setup

- UE3
 - Copy and paste template project
 - .INI hell
 - Lot of manual changes in source code to setup custom project
- UE4
 - Simple project wizard
 - Allow to create project from template depending on game genre
 - Separate directory with all project-related code and content
 - Simple settings managing interface for Editor or Project
 - Easy to tweak global or/and platform specific project settings

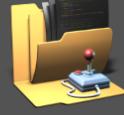
Project Wizard

Unreal Project Browser

Projects New Project Marketplace

Choose a template for your project. If you choose a C++ template, you must have Visual Studio 2013 installed to compile. Blueprint templates use only visual scripting, and do not require additional software.

 A clean empty project with no code.

 Basic Code
An empty project with some basic game framework code classes created.

 First Person
A project that features a first person perspective camera, with a movable player character and an example projectile weapon.

 First Person
A project that features a first person perspective camera, with a movable player character and an example projectile weapon.

 Flying
A project that features a simple flying movement.

 Flying
A project that features a simple flying movement.

Include starter content

Name

Project Settings

Editor Preferences Project Settings

Game

- Description
- Maps & Modes
- Movies
- Packaging
- Supported Platforms

Engine

- Audio
- Collision
- Console
- Cooker
- General Settings
- Input
- Navigation Mesh
- Navigation System
- Physics
- Rendering

Platforms

- Android
- iOS
- Windows

Plugins

- Paper 2D
- Slate Remote
- UDP Messaging

Game - Description

Descriptions and other information about your game.

These settings are always saved in the default configuration file, which is currently writable.

Search

Publisher

Company Name
Homepage
Support Contact

About

Project Thumbnail

Single button arcade game
(4FAAADAA9-45A4-1363-F22A-B5B4C95E7A7C)
Tappy Chicken
1.0.0

Legal

Copyright Notice
Licensing Terms
Privacy Policy

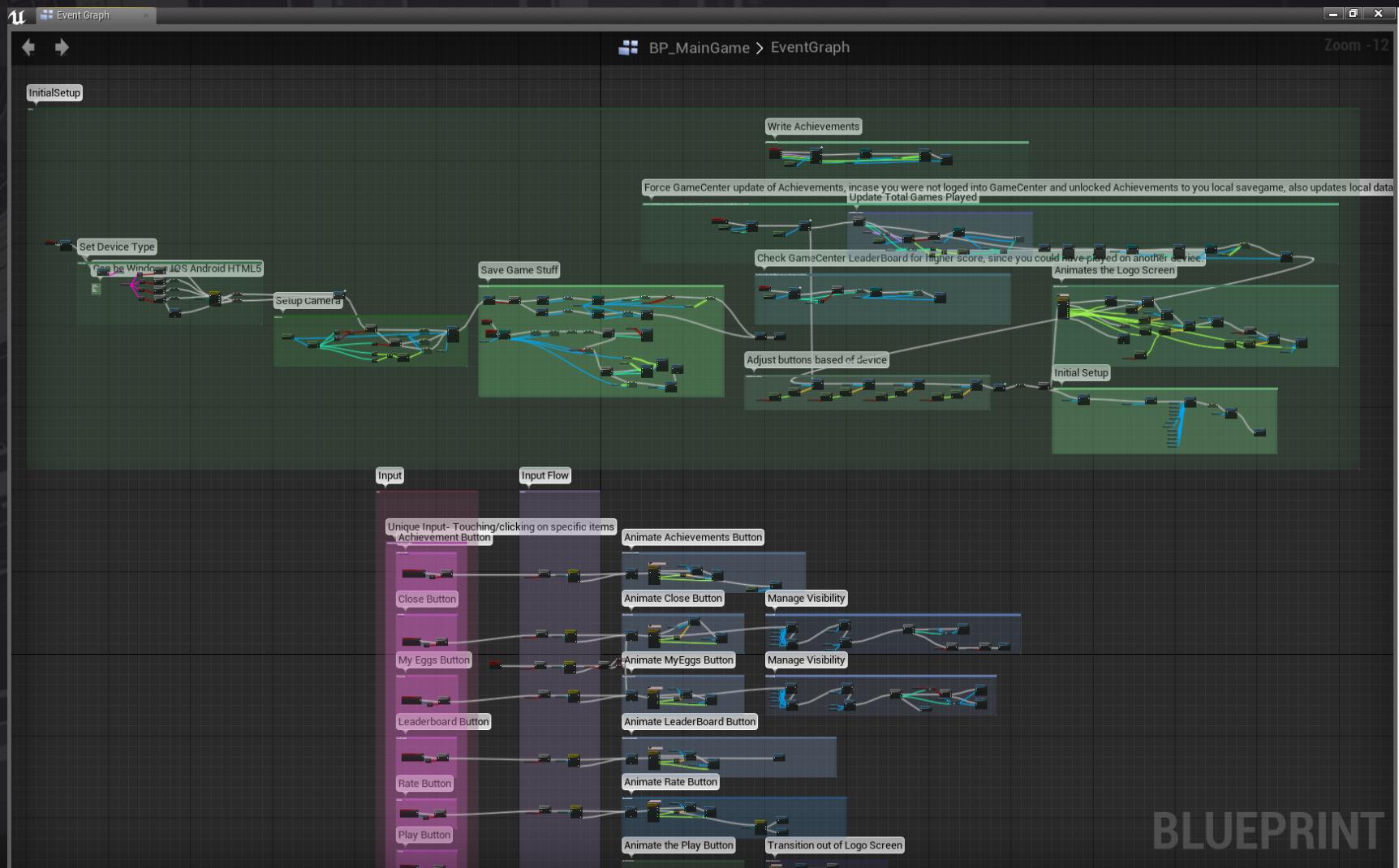
Game Logic Creation (UE3)

- C++ (for full license holders)
 - Was great to have when implementing performance critical gameplay features
 - Was necessary when dealing with low-level systems such as (Animation, Rendering, Physics)
- Unreal Script
 - Java-based language for creating game logic
 - Gives a possibility to implement complex game logic without changes in C++ code
 - Fast iteration times comparing to C++
 - Nearly 10 times slower than C++
- Unreal Kismet
 - Graph-based system for creating high-level game logic
 - Required support from programmers/scripters who deal with Unreal Script
 - Debug was painful

Game Logic Creation (UE4)

- C++
 - Provide low-level control over gameplay aspects
 - Allow to extend basic classes from Gameplay Framework
 - Still optional depending on game project complexity
- Blueprint
 - Derived from Kismet but more powerful and complete
 - Serve many of the same purposes that Unreal Script handled
 - Extending classes
 - Storing and modifying default properties
 - Managing subobject (e.g. Components) instancing for classes
 - Has its own visual debugger

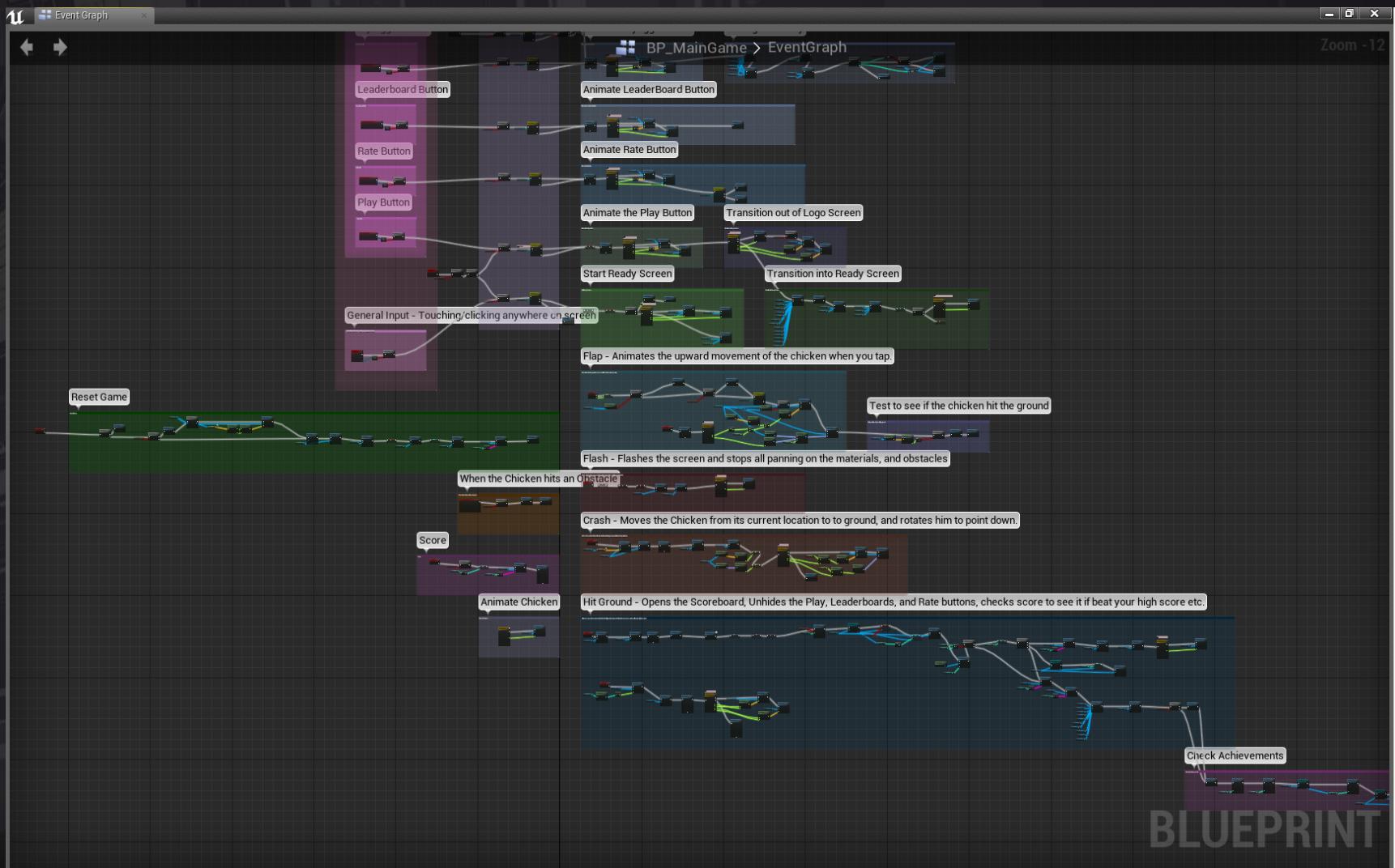
Blueprint Example 1/3



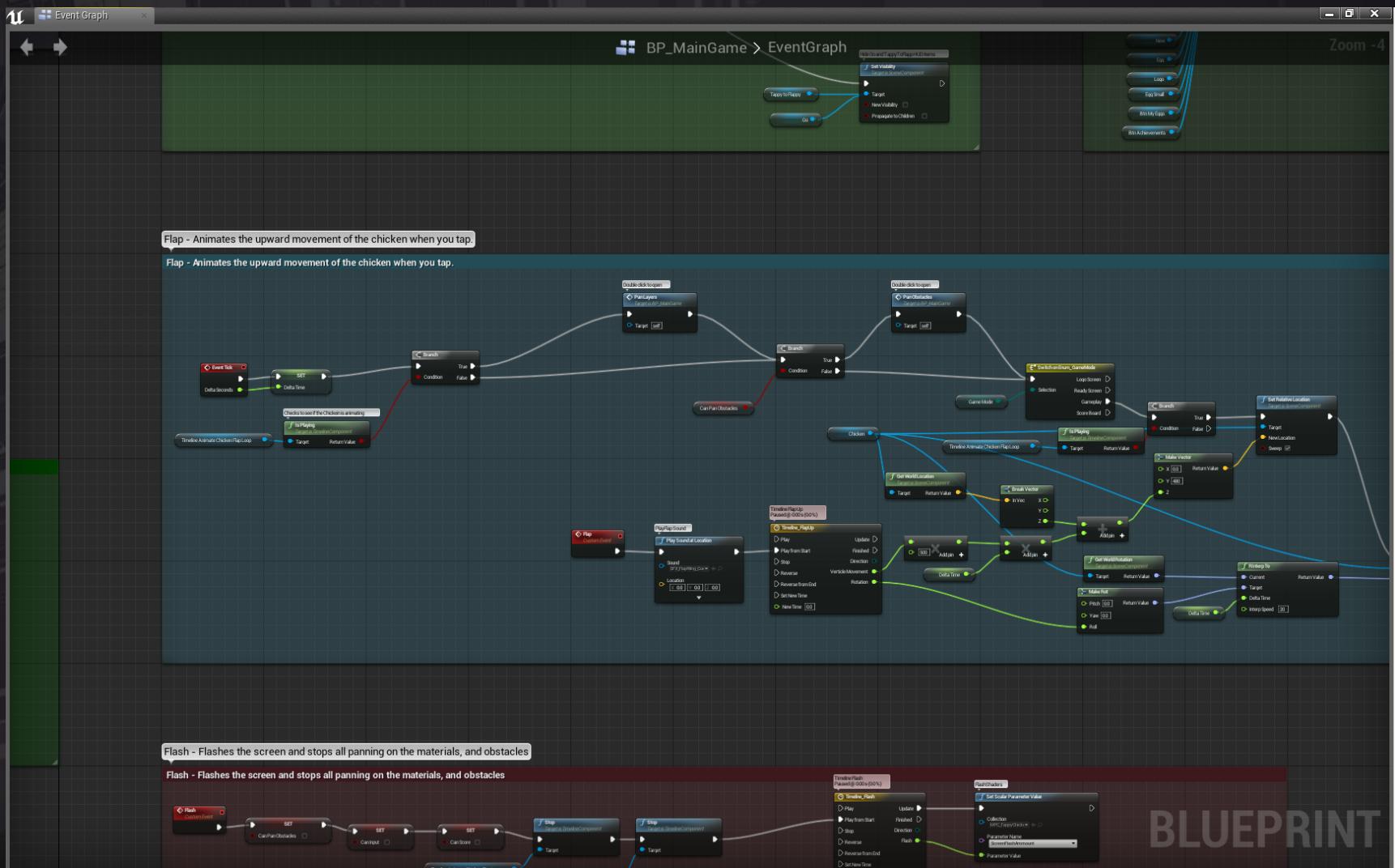
BLUEPRINT

S P E R A S O F T

Blueprint Example 2/3



Blueprint Example 3/3



UI Creation Tools

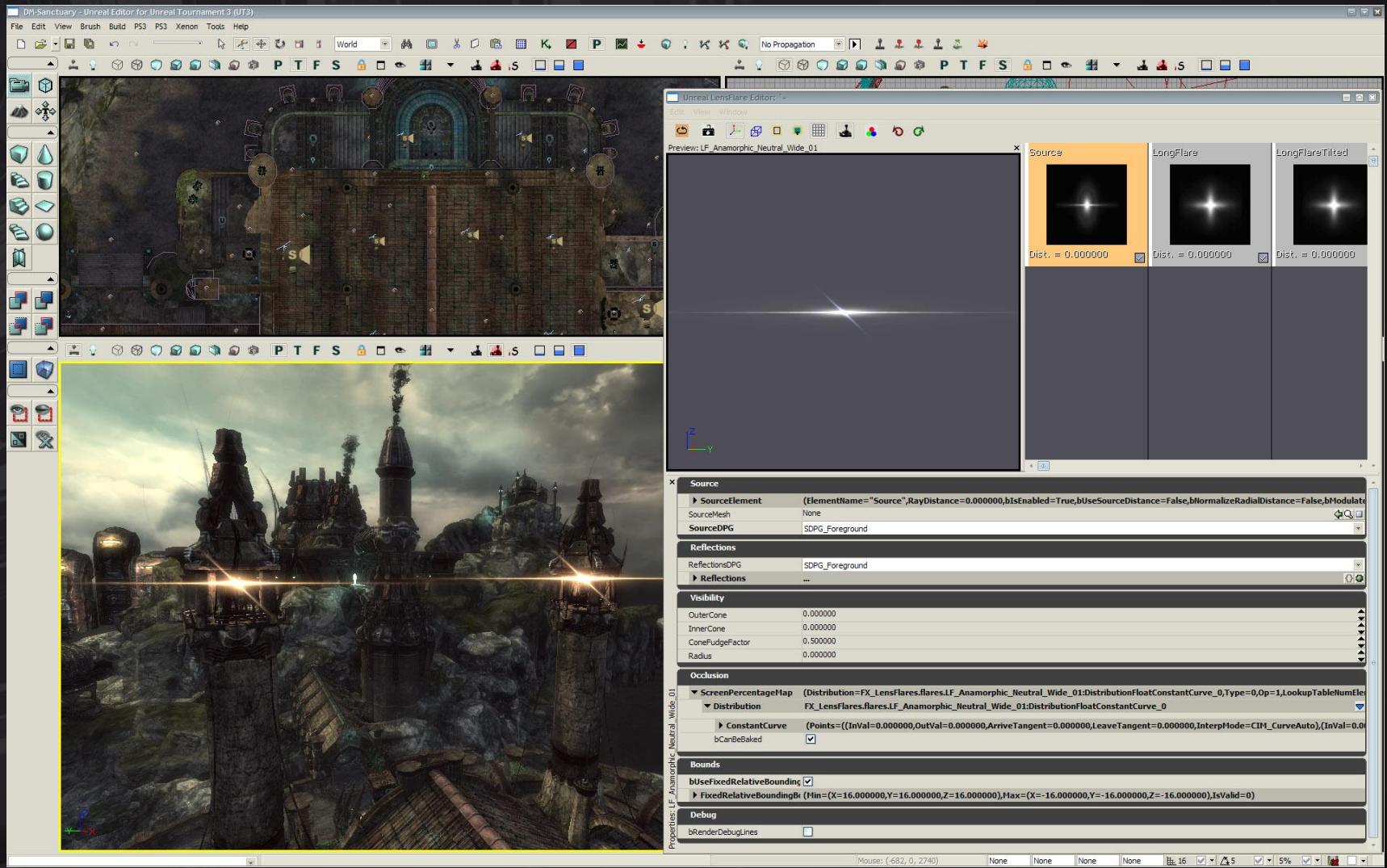
- UE3 – Multiple tools/technologies for similar problems
 - Scaleform for creating ingame UI
 - Custom UI framework for creating UI for mobile platforms
 - The Editor's UI is built using Windows-based frameworks
- UE4 - Completely new UI system called Slate
 - Fully platform-agnostic UI system
 - Used to create Unreal Editor's UI
 - Can be used to create ingame UI
 - Key features
 - Easy to design, layout and style components
 - Easy to create and iterate on UIs



*Follow us on Twitter
@Sperasoft*

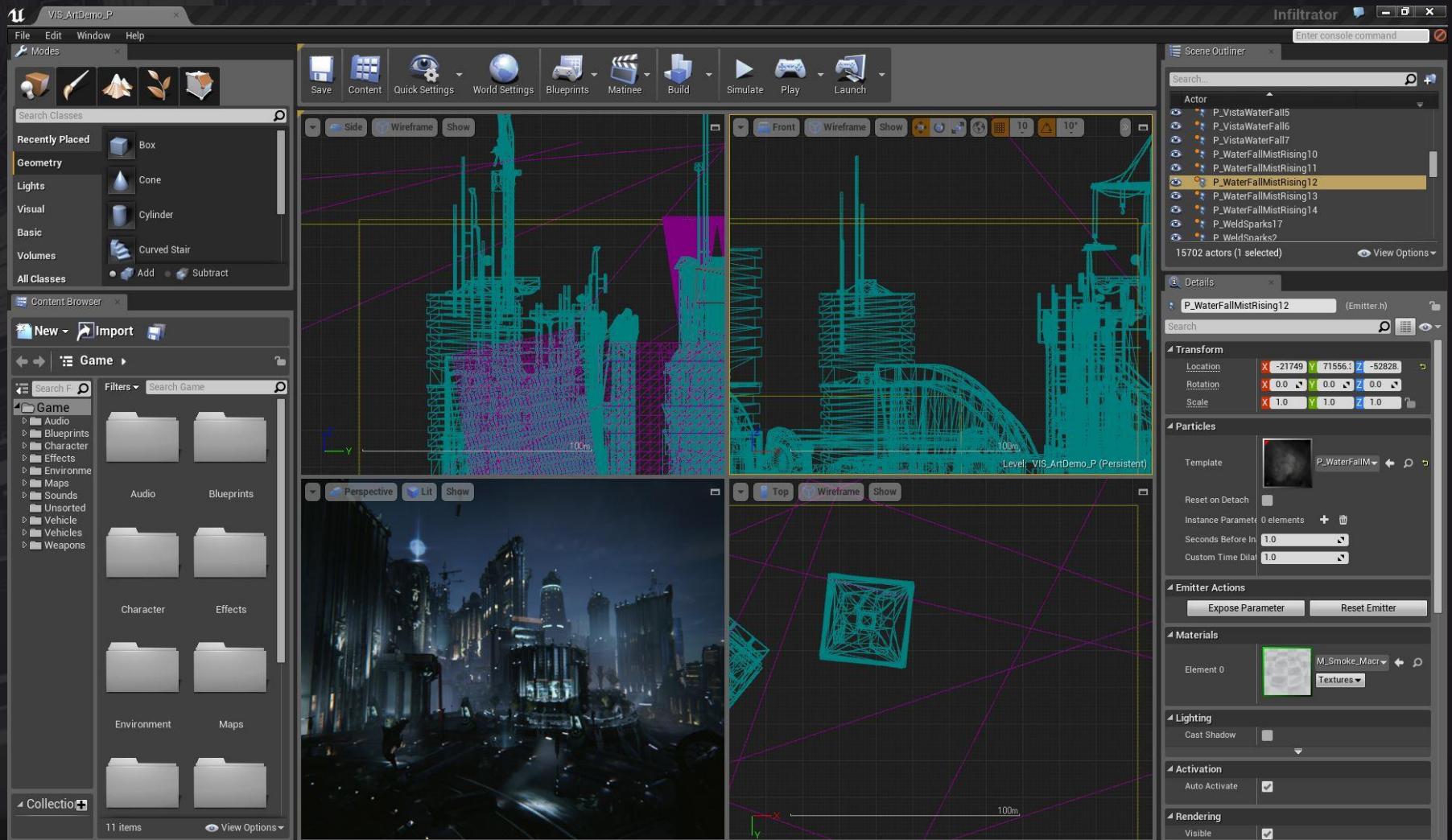
*Visit our site:
sperasoft.com*

Unreal Editor 3 Interface



S P E R A S O F T

Unreal Editor 4 Interface



S P E R A S O F T

Automation System

- Completely new system comparing to UE3
- Provides the ability to perform Unit/Feature testing
 - Unit tests : API-level verification tests
 - Feature tests : System-level verification
 - In-game stats, changing resolution, taking screenshot works
 - Content stress tests
 - Loading all maps, recompiling all blueprints, recompiling all shaders
 - Smoke tests: Unit tests which runs every time in the editor and are considered to be very fast
- Screenshot-based tests
 - Mostly for testing rendering-related errors

Automation Framework Example 1/2

The screenshot shows the Automation Framework interface in a dark-themed application window. The top navigation bar includes tabs for Console, Automation (selected), and Profiler. Below the tabs is a toolbar with icons for Start Tests (27), Refresh Tests, Find Workers, Errors, Warnings, Smoke Tests, Dev Content, and Vis Cmdlet, along with a search bar.

The main area displays a list of test categories under the heading "Test Name". The categories are:

- Core (23) (checked)
- Slate (1) (checked)
- Editor (22)
- Blueprints (353)
- Engine (6) (checked)
- Maps (44)
- QA (5)
- Performance Audits (118)
- OSS (21)

To the right of the list are columns for "Duration" and a progress bar icon. At the bottom of the list, there are three circular icons with question marks.

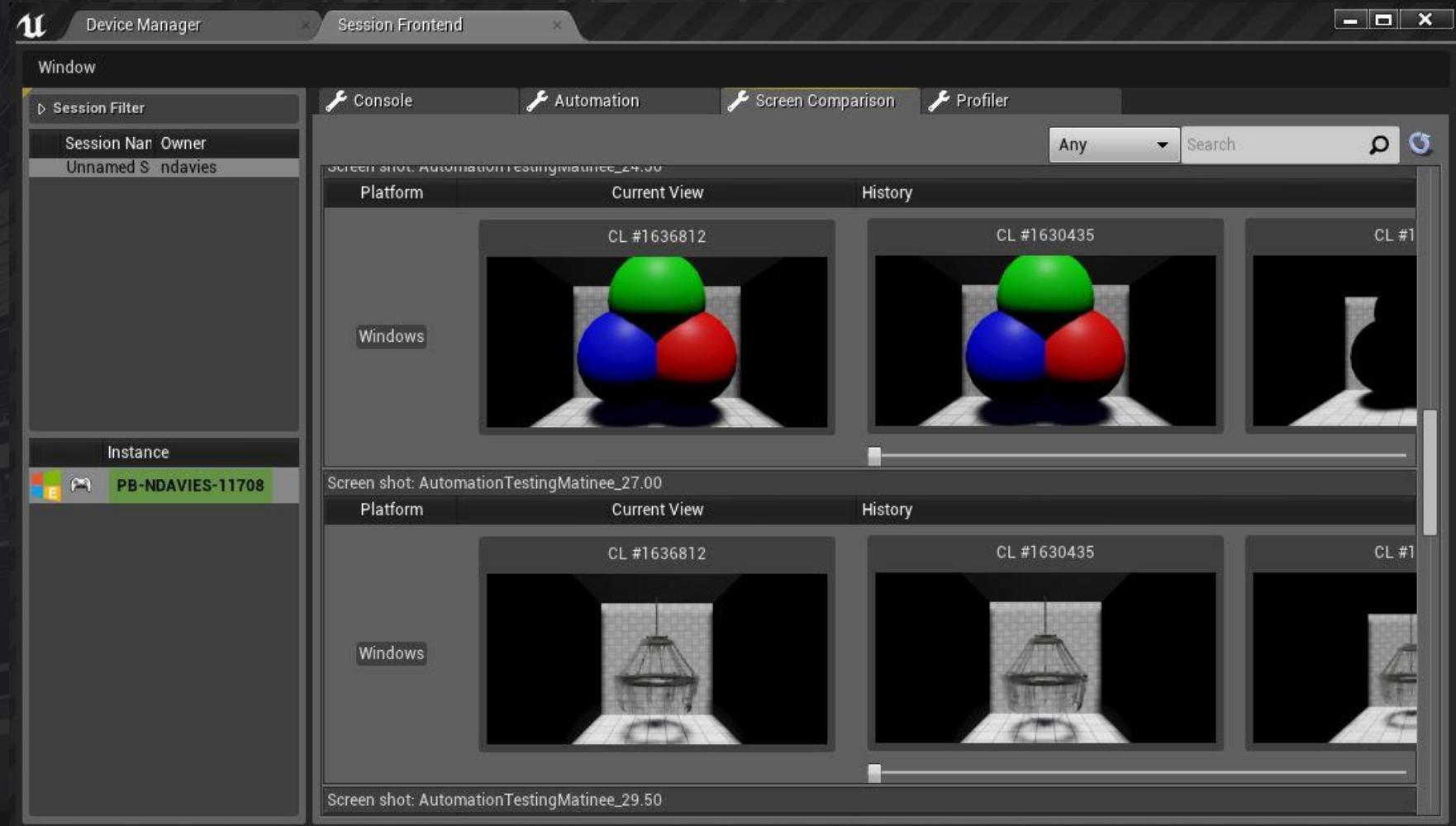
Below the list is a section titled "Automation Test Results:" which is currently empty.

On the far right, a context menu is open with the following options:

- Export All (checked)
- Export Status (checked)
- Export Errors (checked)
- Export Warning (checked)
- Export Logs

Below the menu are buttons for "Export Data" (highlighted in blue), "Export", and "Copy".

Automation Framework Example 2/2



Physically Based Materials

- New Physically Based Shading model
 - Approximates what light does, not what we intuitively think it does
 - Result – more accurate and natural looking
 - Simple properties
 - Base Color
 - Specular
 - Roughness
 - Metallic
 - Values for that properties could be measured in real-world materials
 - Material blending/layering is simple
 - Hard to create physically implausible materials

Mobile Development with UE4

Supported Platforms

- Unreal 4 was developed targeting high-end PCs and next-gen consoles
 - But supports both iOS and Android platforms
 - Android support is still in its early stages
 - Full list of supported devices could be found on wiki
 - All tested devices are grouped in feature tiers
 - Example of classifying iOS devices by performance tiers

Device	LDR	Basic Lighting	Full HDR	Full HDR w/ Sun
iPhone4	🟡 Non-Retina	🔴 Unsupported	🔴 Unsupported	🔴 Unsupported
iPhone4S	🟢 Retina	🟡 Non-Retina	🔴 Unsupported	🔴 Unsupported
iPhone5	🟢 Retina	🟢 Retina	🟡 Non-Retina	🔴 Unsupported
iPhone5S	🟢 Retina	🟢 Retina	🟢 Retina	🟢 Retina

Performance Tiers

- LDR (The highest performance tier supported in UE4)
 - No lighting, no postprocess
- Basic Lighting
 - Static lighting and precomputed GI features are available
 - Full HDR pipeline with access to some postprocess features like tonemapping
- Full HDR Lighting
 - Realistic specular reflections on surfaces with support for varying roughness
 - Full support of normal mapped surfaces
- Full HDR Lighting with per-pixel lighting from the Sun
 - Same as prev. tier, but per-pixel diffuse and specular lighting

General Performance Guideline

- Use post process features only with high-end devices
 - Bloom/DOF is enabled by default (as on PC) Try to disable
 - Such features can easily cost 60+ ms
 - Always profile your game on target devices before considering to use PP feature
- Control scene complexity
 - Careful control over DIP count (< 700) and triangles count (< 500k)
 - The areas with poor occlusion are going to be the biggest challenge
 - PVS still helps
 - Take care of proper setup visibility volumes and built visibility before run on device
- Double check materials
 - Try to simplify. Fewer instructions and texture lookups is better
 - Use independent texture fetches. (UVs in pixel shader shouldn't be changed)

Content Creation Guideline

- Materials

- Only 5 texture samplers are allowed due to HW limitations
- Translucent and Masked Materials are extremely expensive. It is recommended that you use opaque Materials wherever possible
- Several features like Tessellation and Refraction and Subsurface Scattering shading model are not available
- Only Default and Unlit shading models are available

- Textures limitations

- Maximum texture size 2048x2048
 - PVR compressor limitation
- Dimension must be a power of 2
 - Memory usage became more efficient on mobile devices

- Mesh limitations

- <= 65k vertices per model
- <= 75 bones per skeletal mesh

Bonus Slide : Documentation

- Full documentation available online
- In-editor tutorials
 - Useful to start working with Unreal tools from scratch
- Samples Library
 - Game samples available online for free
 - Huge source to start Unreal discovering
 - Samples are connected with online documentation
 - Content Example Sample
 - Represents an approach to learn
 - Each level represents a number of stands each of which has its own example asset and doc reference





Have a question?
Like this deck?

Just follow us on twitter
@Sperasoft