Faculty of Mathematics and Physics
Charles University in Prague
30th March 2016

Graphics for Games

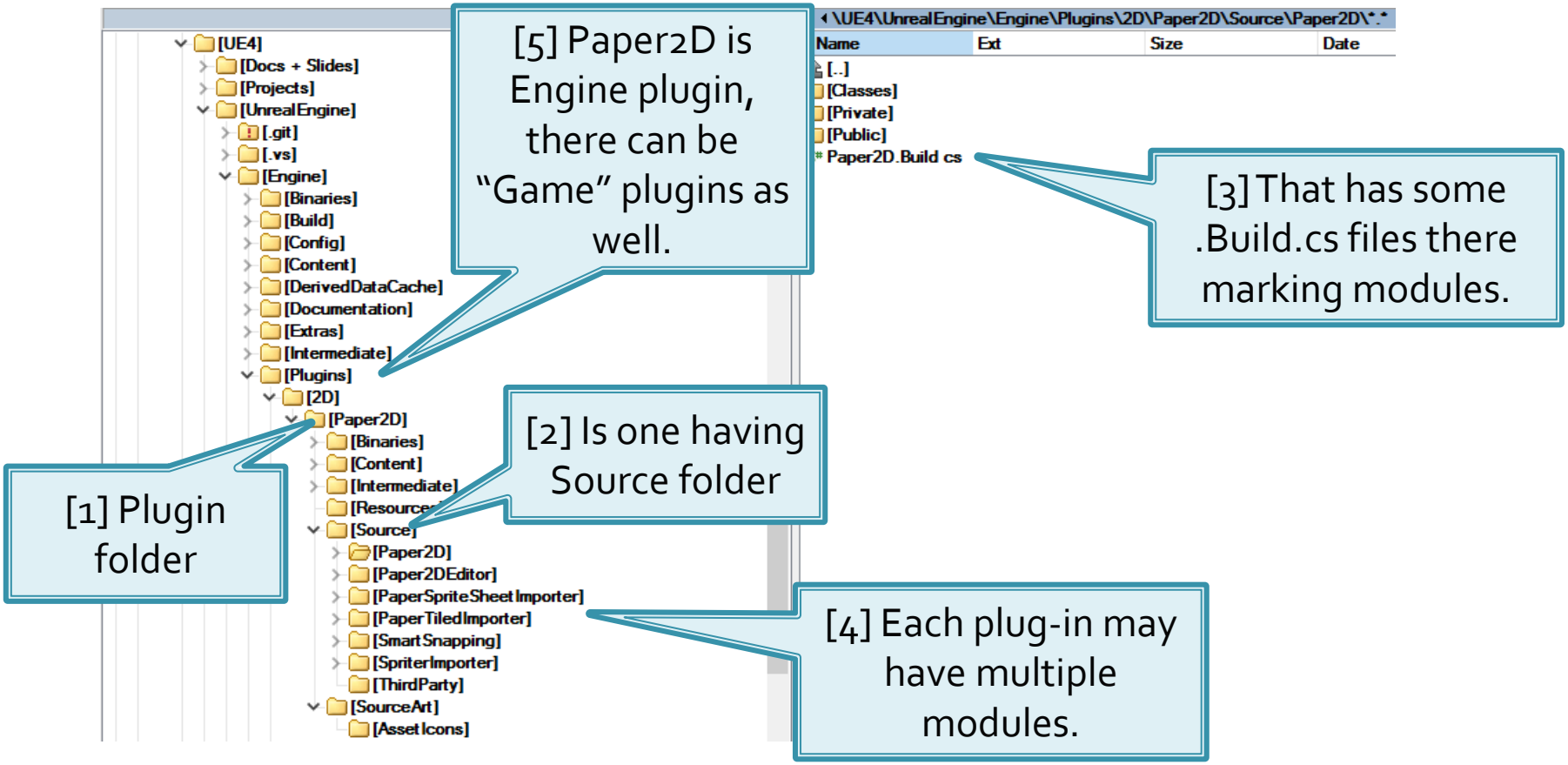
Lab 05 – UE4 – First (2nd) Steps

Resources
Links

- UE4 Programming Introduction
  - https://docs.unrealengine.com/latest/INT/Programming/index.html
- Dos & Don'ts in Coding
  - https://docs.unrealengine.com/latest/INT/Programming/Development/CodingStandard/index.html
- UE4 Plugin Introduction
  - https://docs.unrealengine.com/latest/INT/Programming/Plugins/index.html

Plugins & Modules
Intro

- 1 Plug-in = 1..N Modules

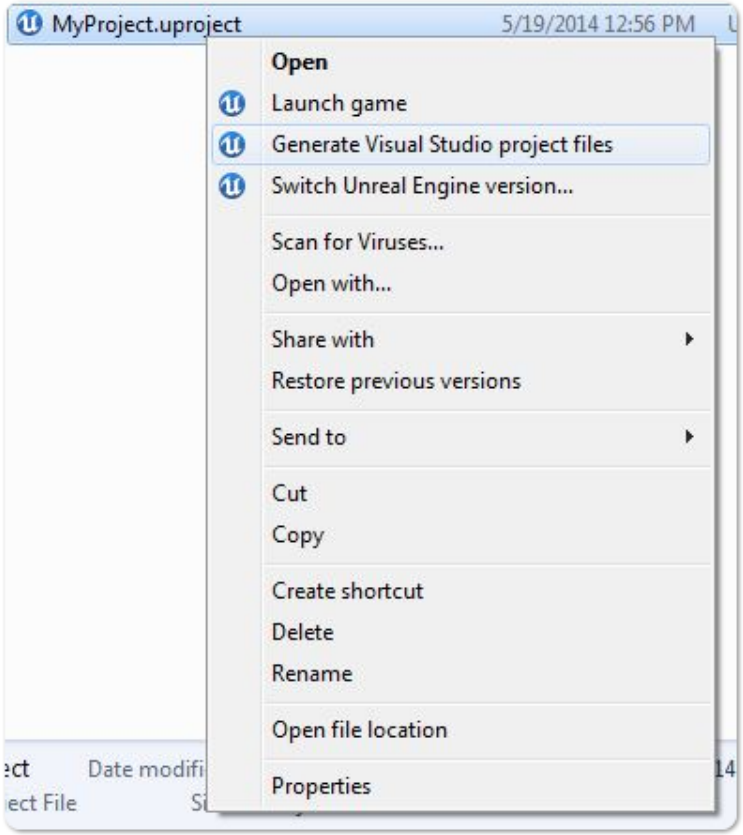


Plugins & Modules

Your own game module

- Your "Project" (your game) always have own module…

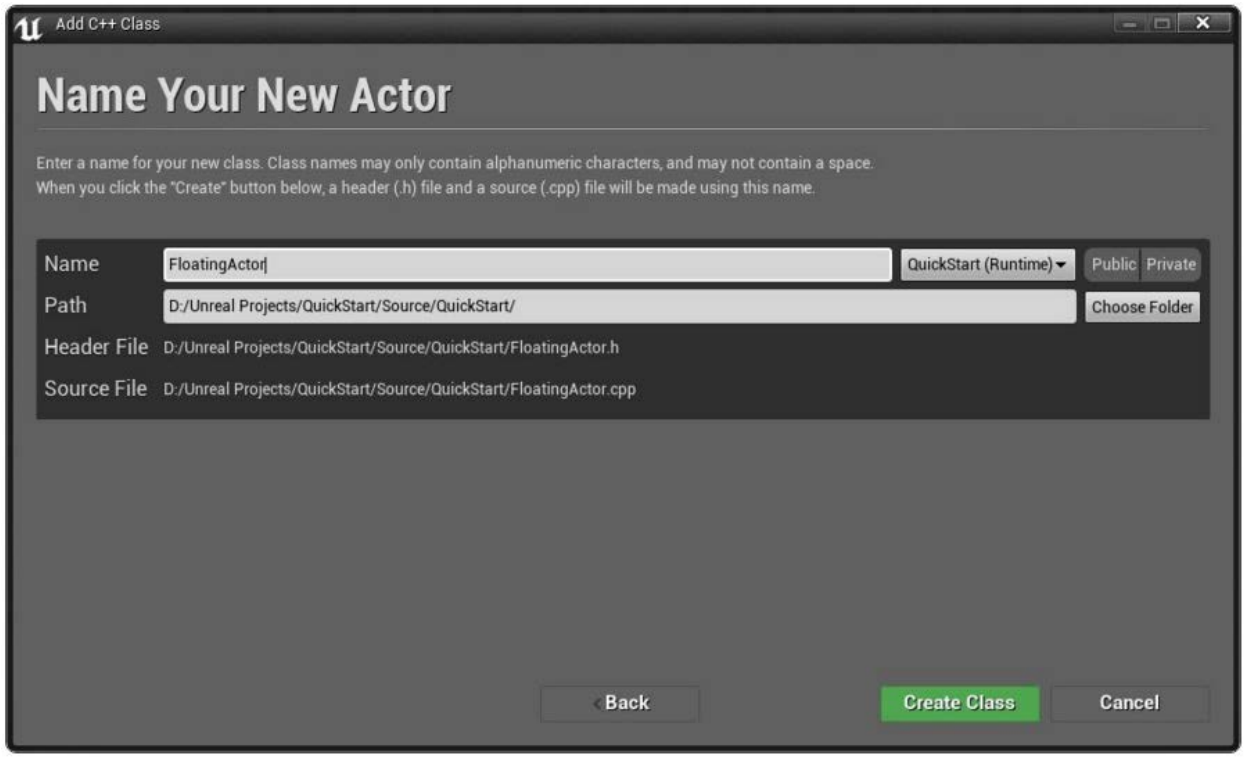
1. Navigate to the location of [ProjectName].uproject in Windows Explorer.
2. **Right-click** on the [ProjectName].uproject file and select **Generate Visual Studio Files**.



Plugins & Modules

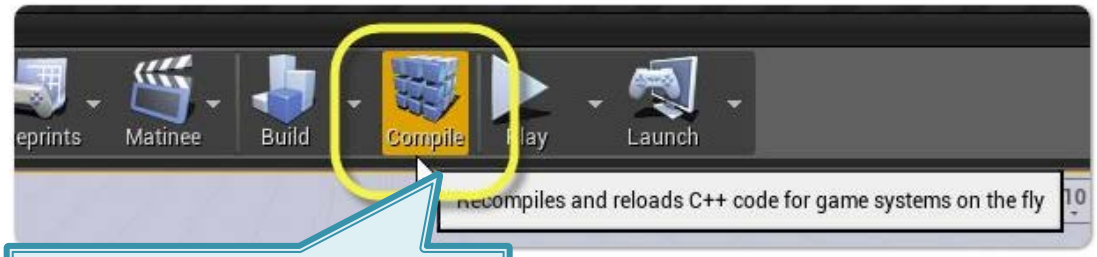
Your own game module

- Check how to setup own classess



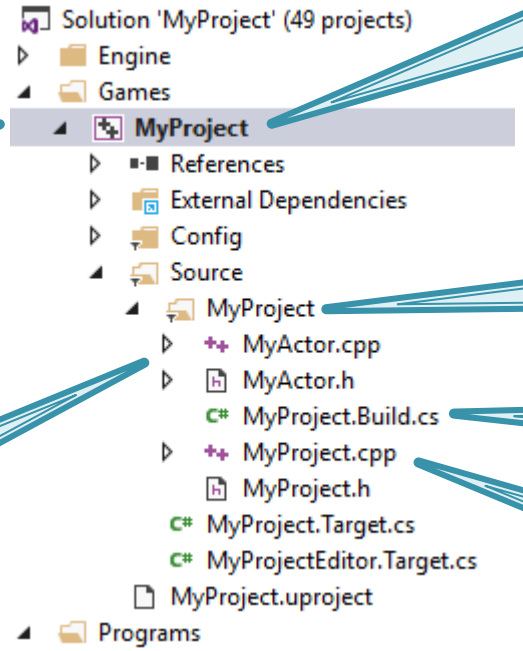
Plugins & Modules

Game module trivia



[7] Or you can trigger compilation from within the Editor

[6] You may build the project to hot reload the code in the editor



[1] Your project folder

[2] Has own Source folder

[3] That has .Build.cs info

[5] Your custom class

[4] And primary module file specifying "MyProject" name

Gameplay Classes
UXxxx

- Building blocks that are gateway to the Unreal Source

VS > Engine > UE4 > Source > Runtime > Engine > Classes > GameFramework

- `UObject`
  - Gameplay base class, provides many services like RTTI
- `AActor`
  - Anything that can be placed within level (even static meshes)
- `UActorComponent`
  - Components that are attachable to actors
- `UStruct`
  - Lightweighted UObject, not GCed (you can use smart pointers), have RTTI, accessible from within editor

Gameplay Classes
UHT Markups (Macros)

- `UENUM()`, `UCLASS()`, `USTRUCT()`, `UFUNCTION()`, `UPROPERTY()`, `GENERATED_UCLASS_BODY()`, `GENERATED_USTRUCT_BODY()`
- Interpreted by UHT (Unreal Header Tool)
- You must let know UHT you are expecting to have your header file processed

  ⇒ Include "not existing yet" header file

  ```
  #include "FileName.generated.h"
  ```

Gameplay Classes
UHT Markups (Macros)

- Markup have arguments (called specifiers in UE4), that may have values

```
UPROPERTY(BlueprintReadOnly, VisibleAnywhere,
          Transient, Category = "Damage")
float DamagePerSecond;
```

- List of UCLASS Specifiers
- List of UPROPERTY Specifiers
- List of UFUNCTION Specifiers
- List of USTRUCT Specifiers

- You can find quick reference within `ObjectBase.h`

Gameplay Classes
Garbage Collection

- Everything that is not reachable from "root set" via UProperty fields is subject for GC
  - RootSet: `YourObjectInstance->SetFlags(RF_RootSet);`
  - Unless you have a reference from plain C++ classes via GC callback
- Instantiate an object:

```
UMyObjectClass* DynamicObj =
        NewObject<UMyObjectClass>(this);
```

- Protect it from GC
  - Header file (.h)

```
UPROPERTY() UMyObjectClass* MyGCProtectedObj;
```

  - Class code file (.cpp)

```
MyGCProtectedObj =
        NewObject<UMyObjectClass>(this);
```

Gameplay Classes
Garbage Collection

- UObjects may be in the middle of "GC"!
- Always test before dereferencing (create a macro for that…)

```
if (!MyGCProtectedObj) return;
if (!MyGCProtectedObj->IsValidLowLevel()) return;
MyGCProtectedObj->GetName(); // safe
```

Gameplay Classes
Garbage Collection

- Referencing UObjects from Non-UObject classes

```
class FMyNormalClass : public FGCObject {
  public:
    UObject* SafeObject;

    FMyNormalClass(UObject* Object) :
      SafeObject(Object) { }

    void AddReferencedObjects(
      FReferenceCollector& Collector) override
    {
      Collector.AddReferencedObject(SafeObject);
    }
};
```

Gameplay Classes
Refactoring

- Check Answer Hub
- DefaultEngine.ini

```
[/Script/Engine.Engine]
+ActiveGameNameRedirects=
   (OldGameName="/Script/OldName",
    NewGameName="/Script/NewName")
```

HANDS ON TIME!

Step-by-step

Creating Custom Actor
Step 1

- Follow this tutorial:
  - https://docs.unrealengine.com/latest/INT/Programming/QuickStart/index.html

1. Learn how to create custom actors and expose properties for your code

Using Blueprints
Step 2

- Read this:
  - https://docs.unrealengine.com/latest/INT/Engine/Blueprints/GettingStarted/index.html

1. Use Level Blueprint to instantiate 8 actors (you have created in previous step) at the beginning of the game in a circle around some point
   - You will need info how to create subobjects from within C++ file: https://docs.unrealengine.com/latest/INT/Programming/Tutorials/Components/1/index.html

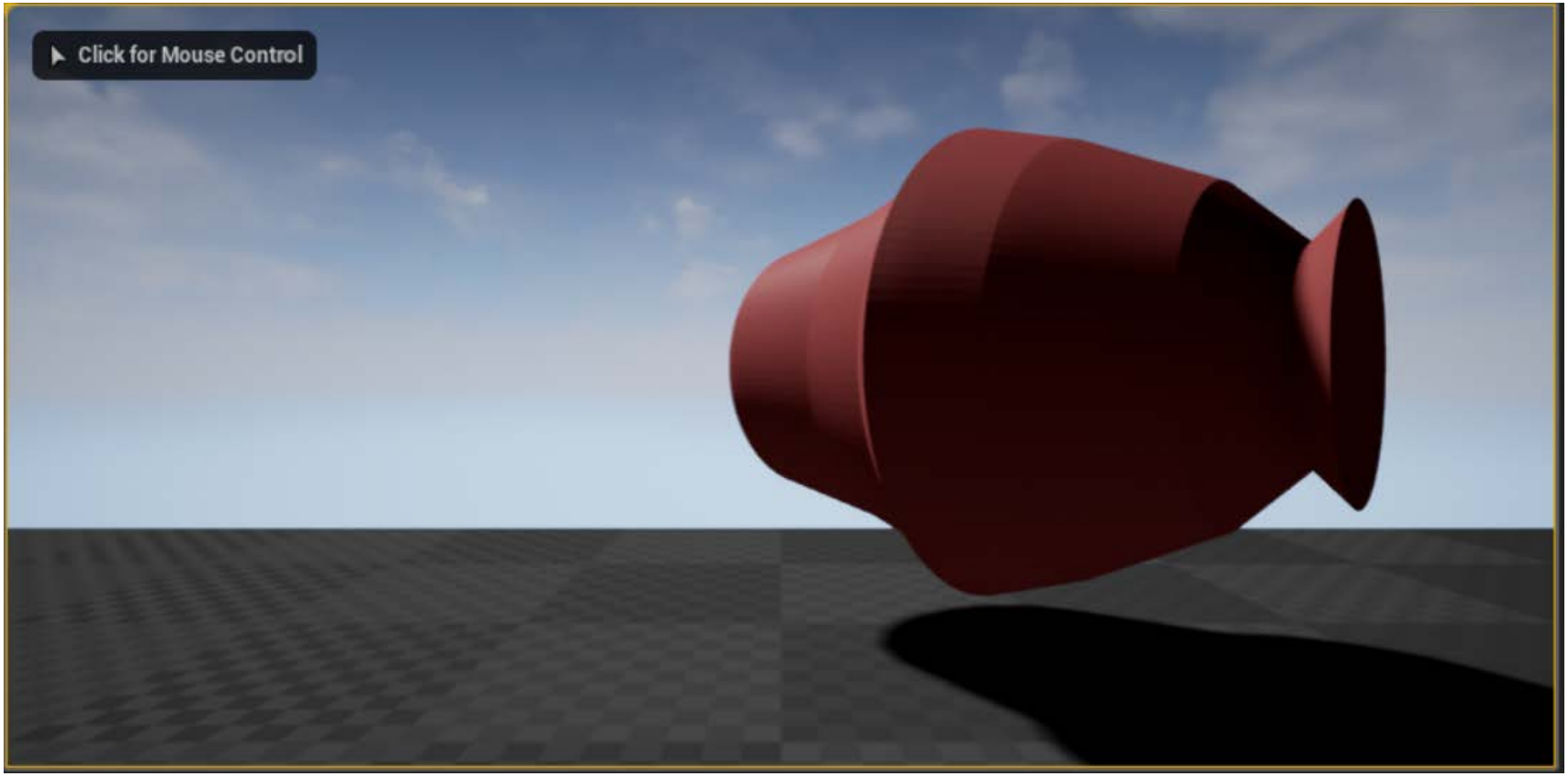
Beneath the Surface
Step 3

- Download "MyProceduralMesh" project:
  - http://artemis.ms.mff.cuni.cz/gemrot/lectures/gcg/2016/UE4MyProceduralMesh-4.11-FixIt.zip

1. Fix it so it is loadable & compilable
2. Use Level blueprint to instantiate MyProceduralLatheActor at the beginning of the game

Beneath the Surface
Step 3

- The result should look like this…



Playing with the Example
Step 4

- Continue with “MyProceduralMesh” project:
  - https://docs.unrealengine.com/latest/INT/Engine/Blueprints/GettingStarted/index.html

1. Make MyProceduralLatheActor usable from the editor as actor (placable into the level)
2. Provide a way for customizing points for PCG from within the editor
3. Provide a way for customizing the material
4. Create own material and assign it to the actor