

Plugins in Unreal Engine 4

Keegan Gibson

Digital Confectioners



Kingsmen

What kind of game are we creating?

- Turn based tactics
- Units could move around the level up to a max movement range
- Units could take cover behind the environment
- Units could climb up the environment

Navigation Requirements

- Units should move in a grid
- Automatic Navigation grid building.
- Special Grid nodes for Ladders
- Varied terrain height

Does it already exist?

- Unreal Engine Nav Mesh
 - Already Exists
 - Automatically generates
 - Actors can path across it

Where do we begin?

What is a plugin?

- Self contained set of Modules
- Expand Unreal Engine Functionality
- Create Custom Tools

Why create a plugin?

- Plugins stand separate from your game
- Makes the engine easy to update
- Reuse across multiple games

What will our plugin require?

- Offline Navigation Grid
- Save as part of the level
- Easy for level designers
- Integrate into existing workflow

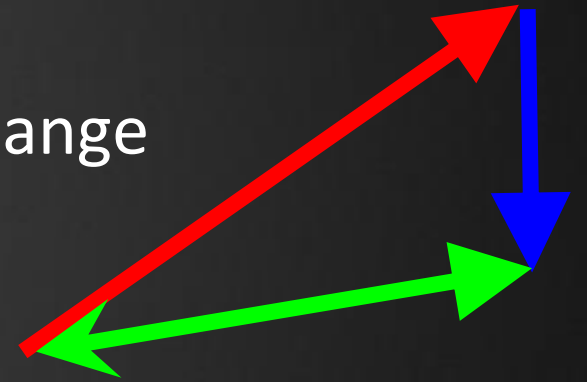
Lets see it in action...

What is a Grid Node?

- Represents a location in the grid where an actor can stand / move through
- Location
- Link to neighbouring nodes
- The actor that is occupying this node

Creating The Grid

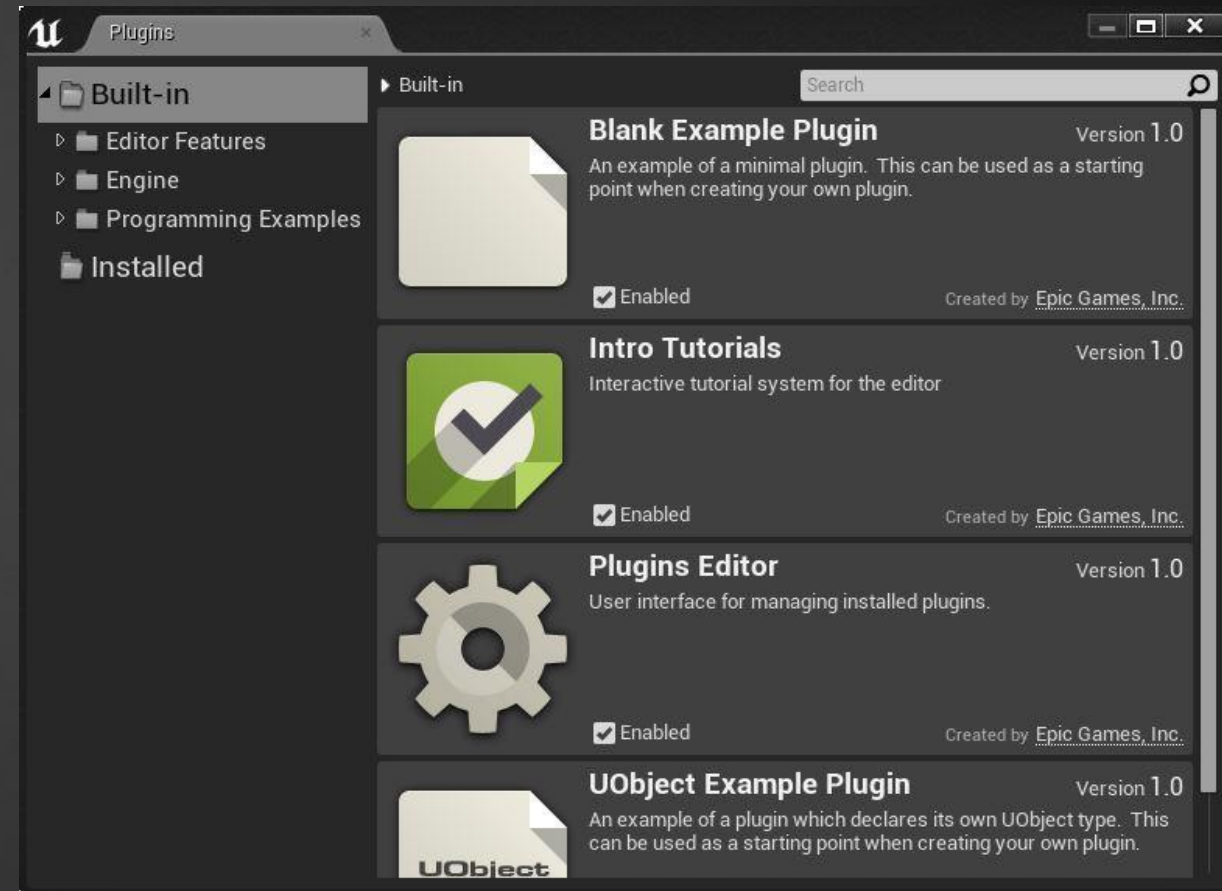
- Initial Start Node
 - Tracing out and up into the world the max height change
 - Traces down to find the location in the world
 - Creates a new GridNode
-
- Use Variance Equality in Vector Comparison
 - Can't touch the world from another thread



So lets get started!

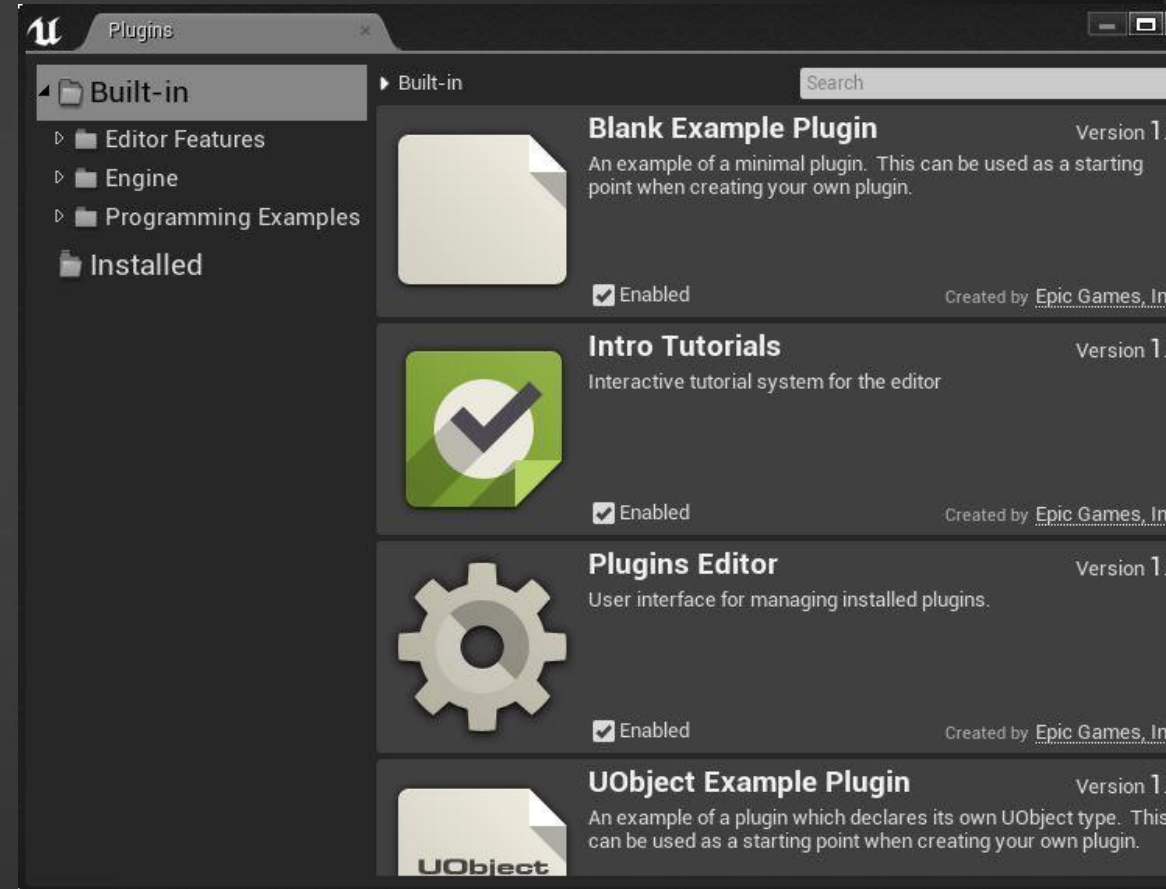
Structure of a Plugin

- Can have Content
- Can have multiple code modules
- .uplugin file descriptors
- Public / Private / Classes



Types of plugins

- Engine vs Installed
 - /UE4 root/Engine/Plugins/My Engine Plugin/
 - /My project/Plugins/My Game Plugin/
- Game Plugins



Where to start?

- Use sample that already exist in the Unreal Engine Folder
 - BlankPlugin
 - UObjectPlugin
- Look at existing plugins
 - /engine/plugins/

Plugin File Descriptors (.uplugin)

```
{
  "FileVersion" : 3,
  "FriendlyName" : "UObject Example Plugin",
  "Version" : 1,
  "VersionName" : "1.0",
  "EngineVersion" : 1579795,
  "Description" : "An example of a plugin which declares its own UObject type.",
  "Category" : "Programming Examples.Plugins",
  "CanContainContent" : "true",

  "Modules" :
  [
    {
      "Name" : "UObjectPlugin",
      "Type" : "Developer"
    }
  ]
}
```

IModule Interface

```
class IGridNavigation : public IModuleInterface
{
public:
    static inline IGridNavigation& Get()
    {
        return FModuleManager::LoadModuleChecked< IGridNavigation >("GridNavigation");
    }

    static inline bool IsAvailable()
    {
        return FModuleManager::Get().IsModuleLoaded( "GridNavigation" );
    }
};
```

Module Implementation

```
#include "IGridNavigationPlugin.h"

class FGridNavigationPlugin : public IGridNavigation
{
    /** IModuleInterface implementation */
    virtual void StartupModule() override;
    virtual void ShutdownModule() override;
};

IMPLEMENT_MODULE(FGridNavigationPlugin, GridNavigationPlugin)

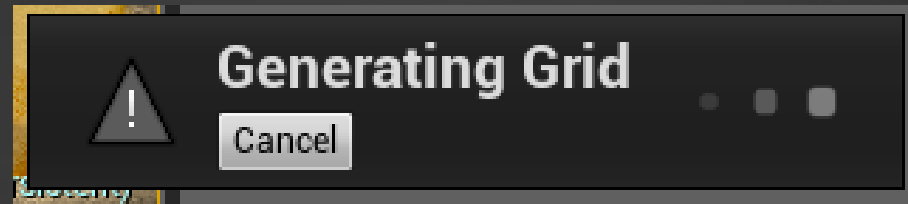
void FGridNavigationPlugin::StartupModule()
{
}

void FGridNavigationPlugin::ShutdownModule()
{
}
```

Working with UE4

- Pop up Notifications
- Settings Menu
- Extend Menus

Creating Notification Popups



```
FNotificationInfo Info("Generating Grid");  
Info.bFireAndForget = false;  
Info.FadeOutDuration = 4.0f;  
Info.ExpireDuration = 0.0f;
```

Adding Buttons To Notification Popups



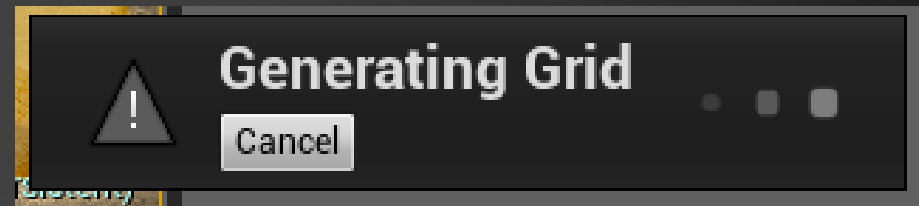
```
// Add a Cancel Button
Info.ButtonDetails.Add(
    FNotificationButtonInfo("Cancel",
        FText::GetEmpty(),
        FSimpleDelegate::CreateUObject(this, &AGridDebug::Clear)));

// Register with Notification Manager
TWeakPtr<SNotificationItem> GridGenerationNotification
    = FSlateNotificationManager::Get().AddNotification(Info);
```

Setting Completion States

```
GridGenerationNotification.Pin()->SetCompletionState(SNotificationItem::CS_Pending);
```

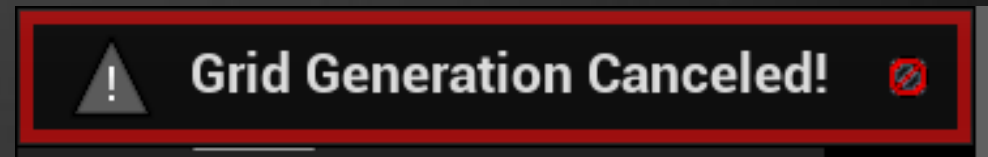
CS_Pending



CS_Success



CS_Fail

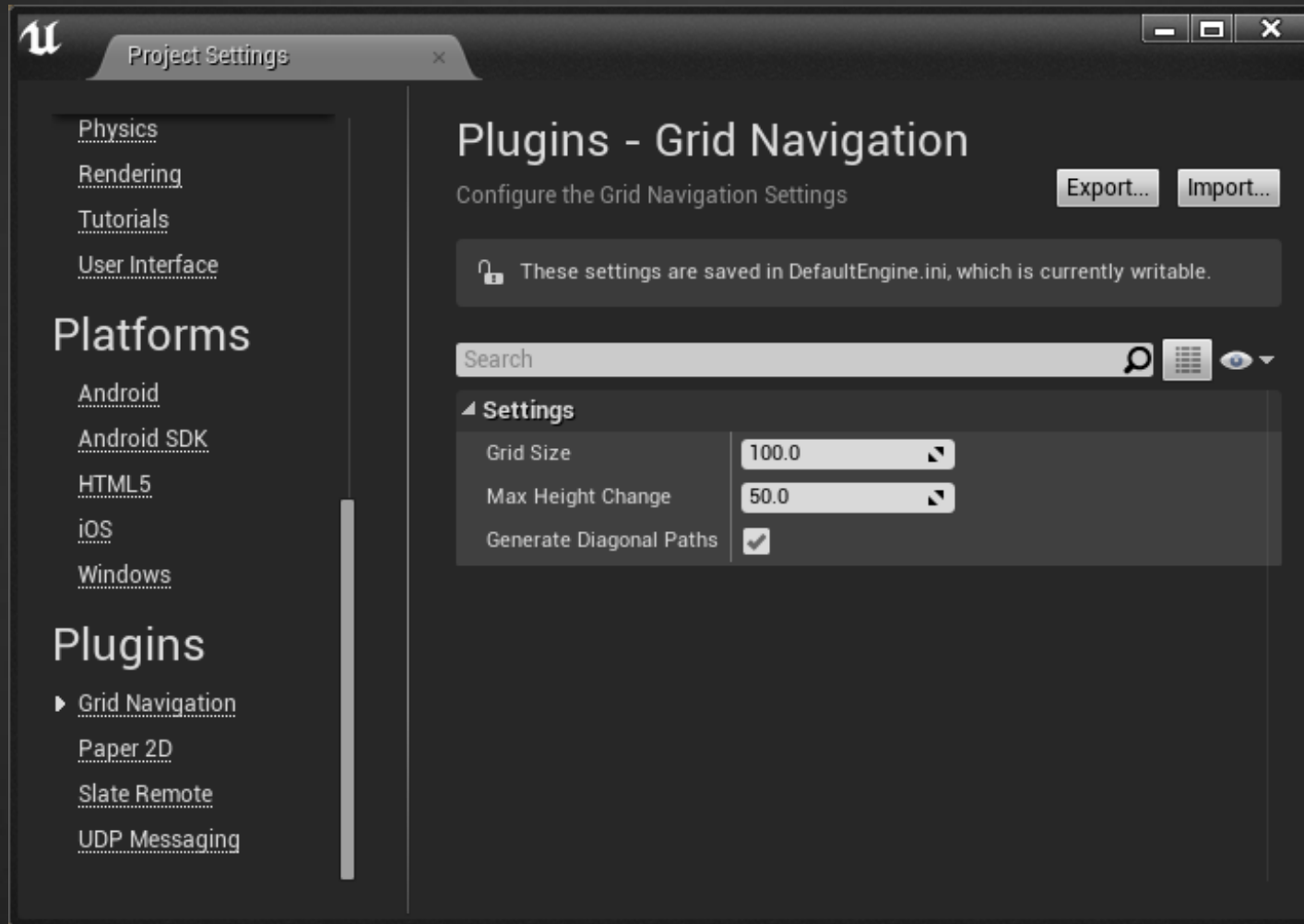


Clearing the Notification

```
NotificationItem->SetText("Grid Finished Generating!");  
NotificationItem->SetCompletionState(SNotificationItem::CS_Success);  
NotificationItem->ExpireAndFadeout();
```



Plugin Settings



Creating Your Own Plugin Settings

```
UCLASS(config = Engine, defaultconfig)
class UGridNavigationSettings : public UObject
{
    // Member Variables
public:
    // The Size of each grid node
    UPROPERTY(GlobalConfig, EditAnywhere, Category = Settings)
    float GridSize;
    // How steep the grid is allowed to generate up.
    UPROPERTY(GlobalConfig, EditAnywhere, Category = Settings)
    float MaxHeightChange;
    // If true grids will have diagonal Paths added to them
    UPROPERTY(GlobalConfig, EditAnywhere, Category = Settings)
    bool GenerateDiagonalPaths;
```

Registering with the Settings Module

```
ISettingsModule* SettingsModule = FModuleManager::Get().GetModulePtr<ISettingsModule>("Settings");
SettingsModule->RegisterSettings("Project",
                                "Plugins",
                                "Grid Navigation",
                                "Grid Navigation",
                                "Configure the Grid Navigation Settings",
                                GetMutableDefault<UGridNavigationSettings>()
                                );
```

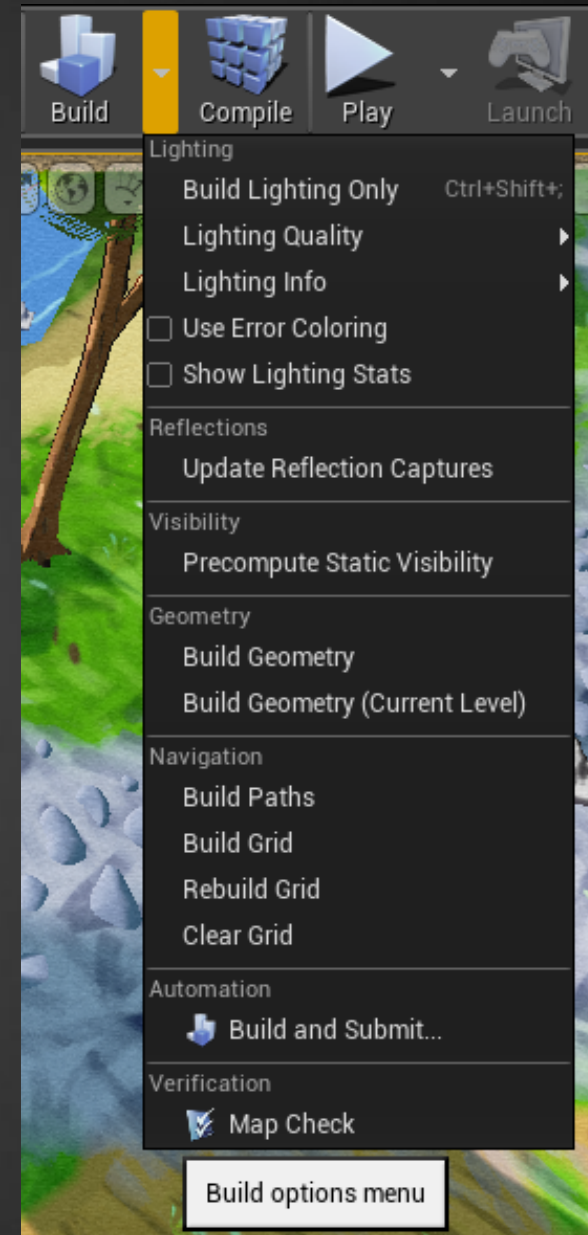
Unregistering with the Settings Menu

```
ISettingsModule* SettingsModule = FModuleManager::Get().GetModulePtr<ISettingsModule>("Settings");  
SettingsModule->UnregisterSettings("Project", "Plugins", "Grid Navigation");
```

Reading your Settings

```
float MaxHeightChange  
    = GetDefault<UGridNavigationSettings>->MaxHeightChange;
```

Extending Editor Menus



Extending Editor Menus - Setup

The screenshot shows the Unreal Engine editor settings interface. On the left is a navigation pane with a gear icon for 'General' and a cube icon for 'Level Editor'. Under 'General', several sub-sections are listed: Appearance, Automation, Region & Language, Keyboard Shortcuts, Loading & Saving, Miscellaneous (which is expanded with a right-pointing triangle), and Experimental. The main area is titled 'General - Miscellaneous' and contains the text 'Customize the behavior, look and feel of the editor.' At the top right of this area are four buttons: 'Set as Default', 'Export...', 'Import...', and 'Reset to Defaults'. Below these buttons is a search bar with a magnifying glass icon and a grid icon. The 'Developer Tools' section is expanded, showing a list of options with checkboxes: 'Display UIExtension Points' (checked), 'Display Documentation Link' (unchecked), 'Display Action List Item Ref Ids' (unchecked), and 'Show Frame Rate and Memory' (unchecked). A tooltip box is positioned over the 'Display Documentation Link' checkbox, containing the text: 'If enabled, any newly opened UI menus, menu bars, and toolbars will show the developer hooks that would accept extensions'.

General

- Appearance
- Automation
- Region & Language
- Keyboard Shortcuts
- Loading & Saving
- Miscellaneous
- Experimental

Level Editor

General - Miscellaneous

Customize the behavior, look and feel of the editor.

Set as Default Export... Import... Reset to Defaults

Search

Developer Tools

- Display UIExtension Points
- Display Documentation Link
- Display Action List Item Ref Ids
- Show Frame Rate and Memory

If enabled, any newly opened UI menus, menu bars, and toolbars will show the developer hooks that would accept extensions

LevelEditorLighting
LevelEditorLighting Lighting

- Build Lighting Only Ctrl+Shift+;
- Lighting Quality ▶
- Lighting Info ▶

Use Error Coloring

Show Lighting Stats

LevelEditorReflections
LevelEditorReflections Reflections

- Update Reflection Captures

LevelEditorVisibility
LevelEditorVisibility Visibility

- Precompute Static Visibility

LevelEditorGeometry
LevelEditorGeometry Geometry

- Build Geometry
- Build Geometry (Current Level)

LevelEditorNavigation
LevelEditorNavigation Navigation

- Build Paths
- Build Grid
- Rebuild Grid
- Clear Grid

LevelEditorAutomation
LevelEditorAutomation Automation

-  Build and Submit...

LevelEditorVerification
LevelEditorVerification Verification

- Map Check

Extending Editor Menus - Widget Reflector



Widget Reflector

Application Scale: 1.0

Show Focus

Widget Name	Fo	Visibility	Widget Info
↳ SBorder	<input type="checkbox"/>	HitTestInvisible	SDockingTabStack.cpp(63)
↳ SOverlay	<input type="checkbox"/>	SelfHitTestInvisible	SDockingTabStack.cpp(145)
↳ SBorder	<input type="checkbox"/>	Visible	SDockingTabStack.cpp(148)
↳ STutorialWrapper	<input type="checkbox"/>	SelfHitTestInvisible	SLevelEditor.cpp(548)
↳ SHorizontalBox	<input type="checkbox"/>	SelfHitTestInvisible	SLevelEditor.cpp(550)
↳ SBorder	<input type="checkbox"/>	Visible	LevelEditorToolBar.cpp(132)
↳ SMultiBoxWidget	<input type="checkbox"/>	Visible	MultiBox.cpp(215)
↳ SBorder	<input type="checkbox"/>	Visible	MultiBox.cpp(682)
↳ SClippingHorizontalBox	<input type="checkbox"/>	Visible	MultiBox.cpp(553)
↳ SVerticalBox	<input type="checkbox"/>	SelfHitTestInvisible	MultiBox.cpp(446)

Extending Editor Menus

```
FLevelEditorModule& LevelEditor =  
    FModuleManager::GetModuleChecked<FLevelEditorModule>("LevelEditor");  
  
// Register the extension with the Level Editor - Build Menu  
BuildMenuExtender =  
    FLevelEditorModule::FLevelEditorMenuExtender::CreateRaw(this,  
        &FGridNavigationEditorPlugin::OnExtendLevelEditorBuildMenu);  
  
LevelEditor.GetAllLevelEditorToolbarBuildMenuExtenders().Add(BuildMenuExtender);
```

Extending Editor Menus

```
TSharedRef<FExtender>
FGridNavigationEditorPlugin::OnExtendLevelEditorBuildMenu(const TSharedRef<FUICommandList> CommandList)
{
    TSharedRef<FExtender> Extender(new FExtender());

    Extender->AddMenuExtension(
        "LevelEditorNavigation",
        EExtensionHook::After,
        NULL,
        FMenuExtensionDelegate::CreateRaw(this, &FGridNavigationEditorPlugin::CreateBuildMenu));

    return Extender;
}
```

Extending Editor Menus

```
void FGridNavigationEditorPlugin::CreateBuildMenu(FMenuBuilder& Builder)
{
    FUIAction Action_BuildGrid(
        FExecuteAction::CreateRaw(this, &FGridNavigationEditorPlugin::BuildGrid),
        FCanExecuteAction::CreateRaw(this, &FGridNavigationEditorPlugin::IsReadyToBuild));

    Builder.AddMenuEntry(
        "Build Grid",
        "Build the Grid, Updates existing Nodes if Possible.",
        FSlateIcon(),
        Action_BuildGrid,
        NAME_None,
        EUserInterfaceActionType::Button);
}
```

What plugins already exist

- Scripting Languages
- Custom Rendering Plugins
- Source Control
- Custom Input Devices
- Editor Functionality

Plugins on the Marketplace

- Phased release of plugins into the marketplace
- Likely in 4.8

Questions?

Thank You!

Keegan Gibson

Keegan.Gibson@digitalconfectioners.com

www.digitalconfectioners.com