

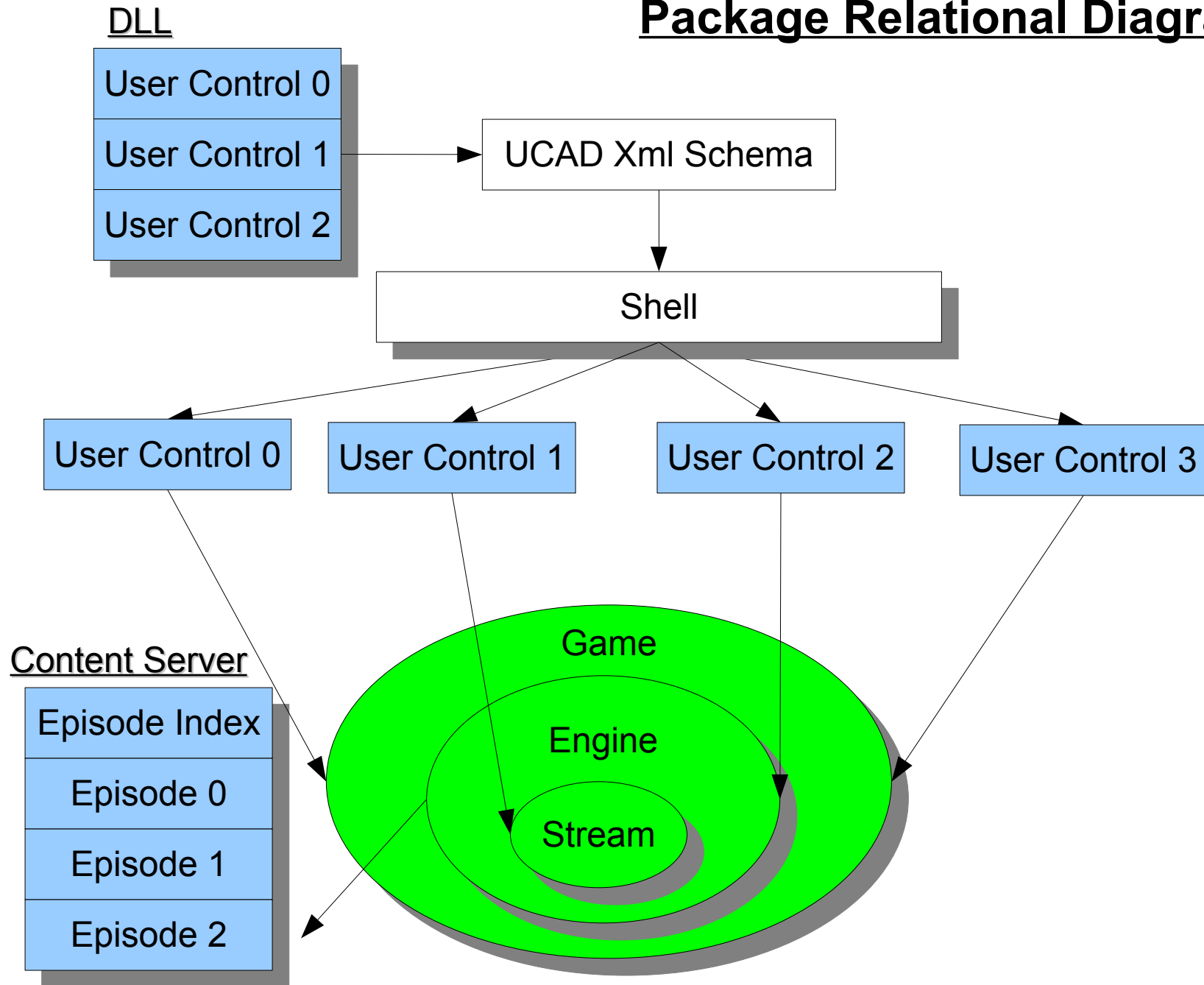
DarkWynter
Virtual Environment

System Architecture Diagrams

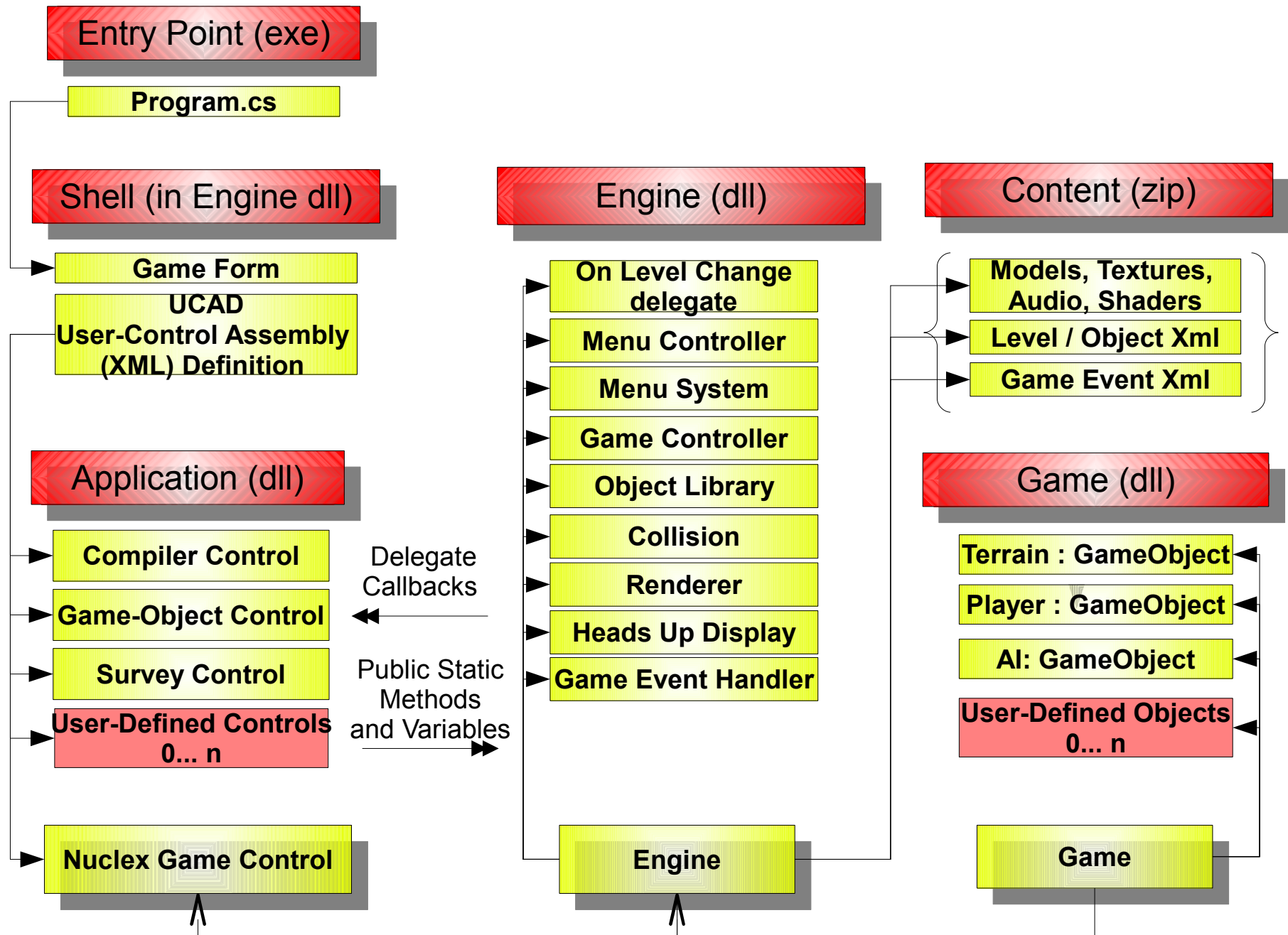
Table Of Contents

- 1) Project Relational Diagram
- 2) Controllers
- 3) Menu System
- 4) Heads Up Display
- 5) Game Objects
- 6) Event System
- 7) Renderer
- 8) Compiler Challenges

Package Relational Diagram



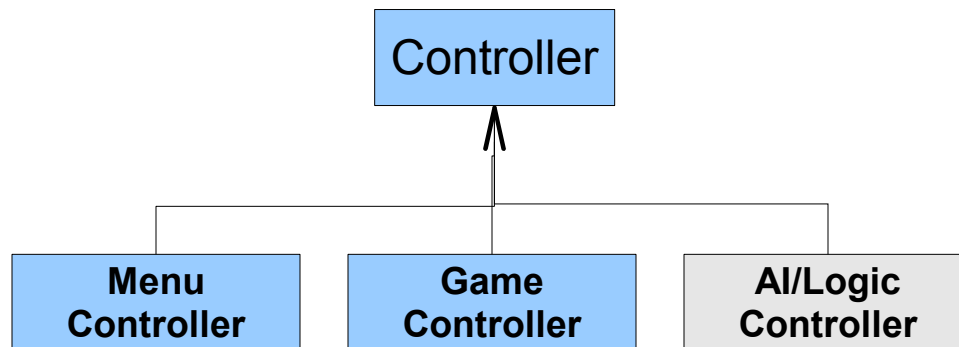
Component Relational Diagram



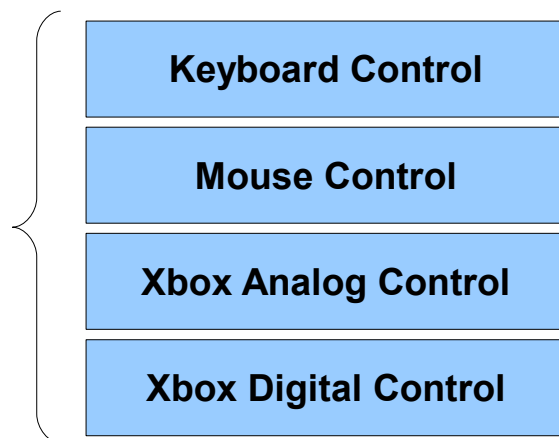
Logical Controllers

Add call associates Keys with Delegate functions.

Any combination of Hardware Controls can be included in a Logical Controller.



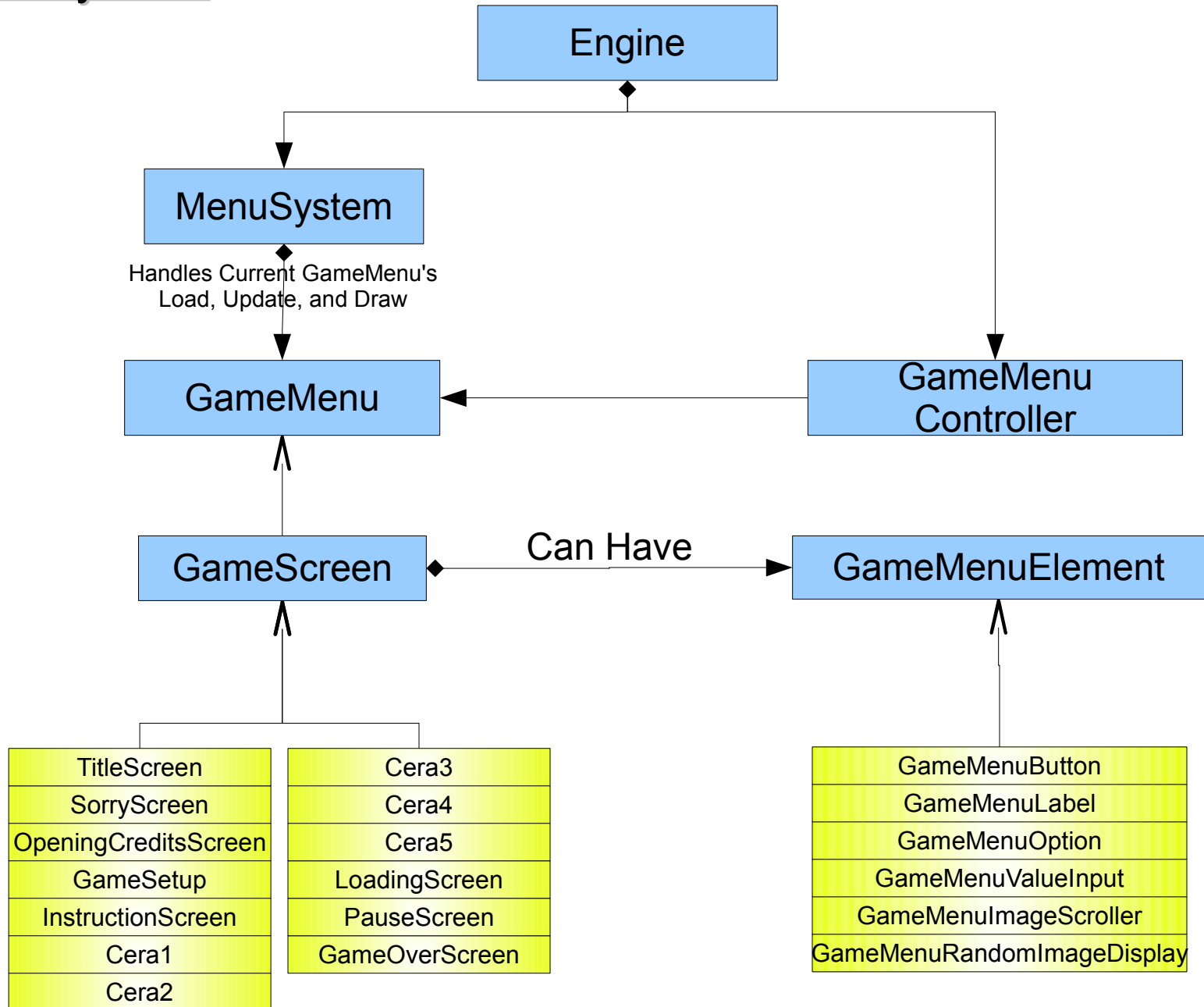
Hardware Controls



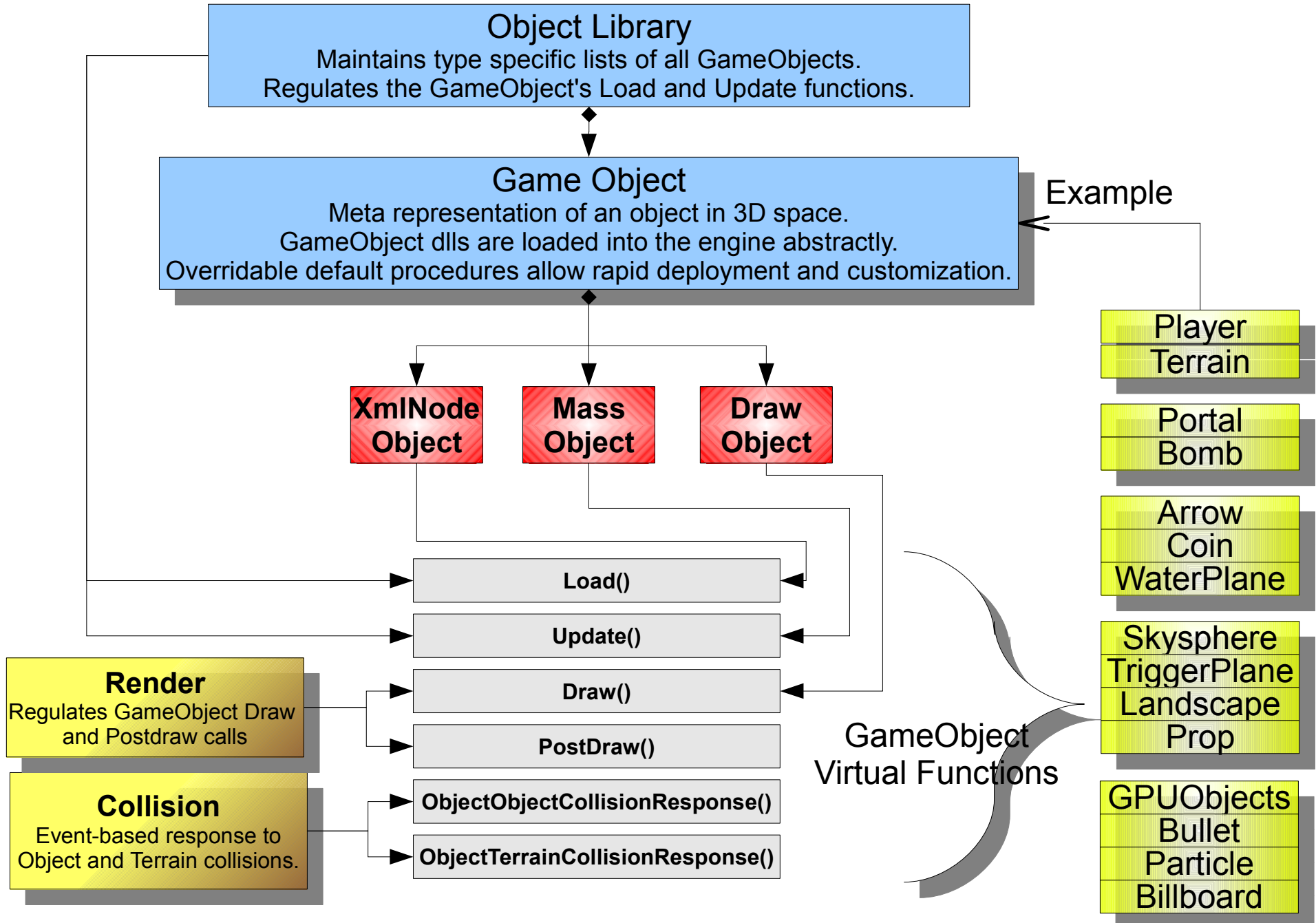
```
// Associate Keys with Delegate functions, using keytimer to control input refresh rate.  
public GameController(int playerNumber): base(playerNumber)  
{  
    Add(new KeyboardControl(Keys.W, MoveForward, 10));  
    Add(new MouseControl(Enums_Engine.ControlType.Motion, Rotate, 0));  
}
```

```
private void MoveForward(ControllerBoolEventArgs args)  
{  
    args.objectLibrary.humans[playerNumber].Translate(new Vector2(0, 10));  
}  
private void Rotate(ControllerVec2EventArgs args)  
{  
    args.objectLibrary.humans[playerNumber].Rotate(args.value);  
}
```

Menu System



Game Objects

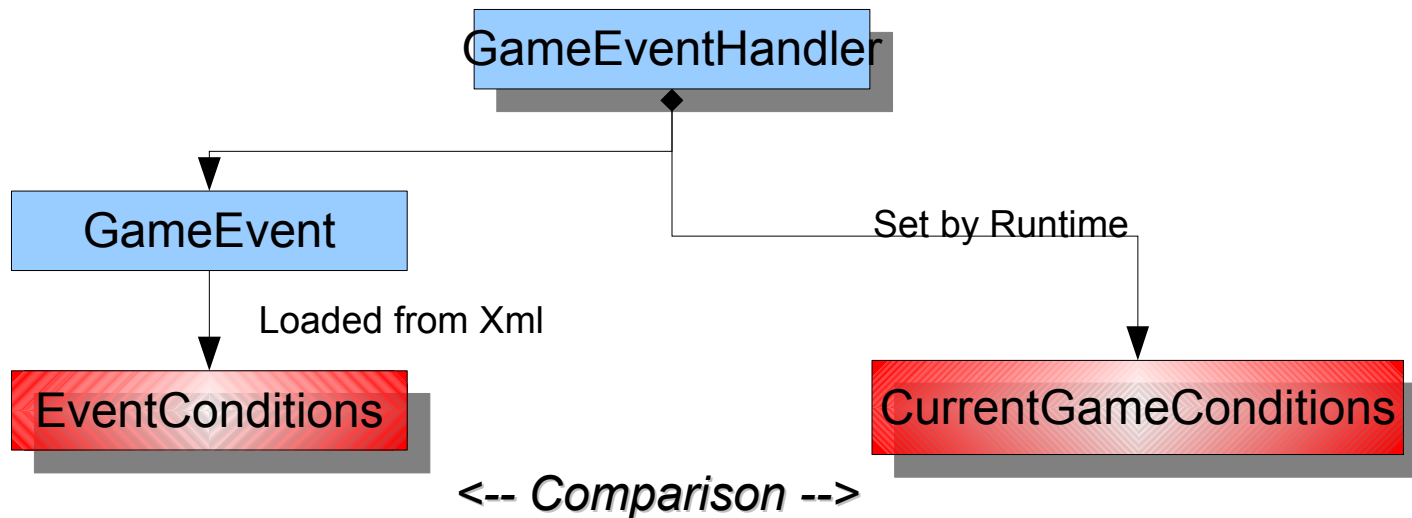


Event System

Events are triggered by modifying the public static CurrentGameConditions .

Updating CGC conditions auto-checks potential matches between CGC and the Xml-defined GameEvents..
(aka - Static Board Evaluation)

The Event System fires matched events, then removes them from the list.



Game-Event Types

DialogueEvent	Updates the HUD with the new dialog
AIEvent	Sets the new position for the AI to go to
TerrainEvent	Builds the terrain between a predetermined set of points
StackEvent	Builds the stack on the HUD based on Player.mass.CurrentPosition
HUDMapEvent	Updates the location map on HUD to reflect Nodes traveled
LevelChangeEvent	Drops events and reloads level

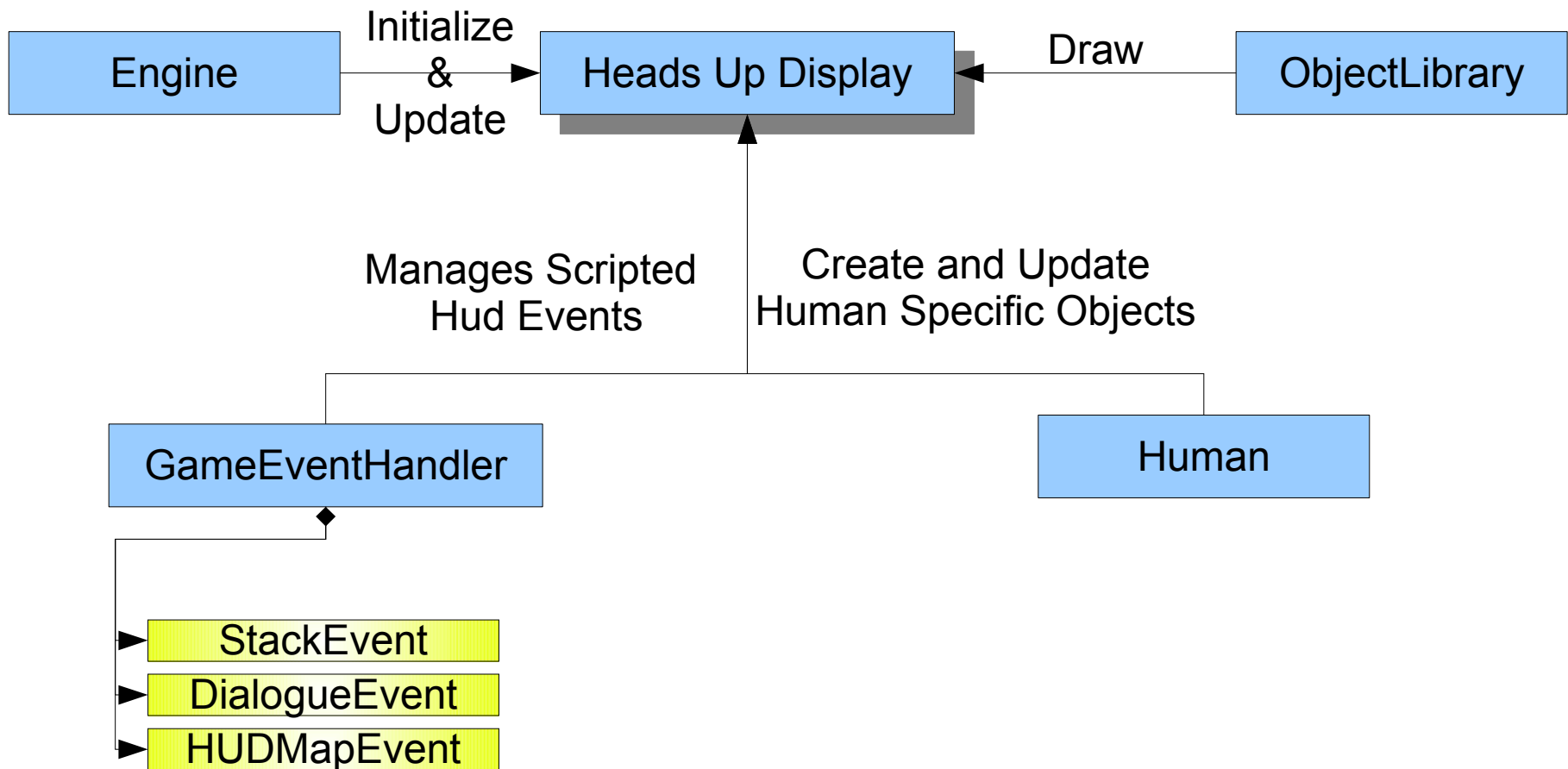
Heads Up Display

Engine initializes and drives HUD updates through static accessors.

Human creates HUD and updates it's own elements.

GameEventHandler initializes and handles updates for scripted HUD Events,

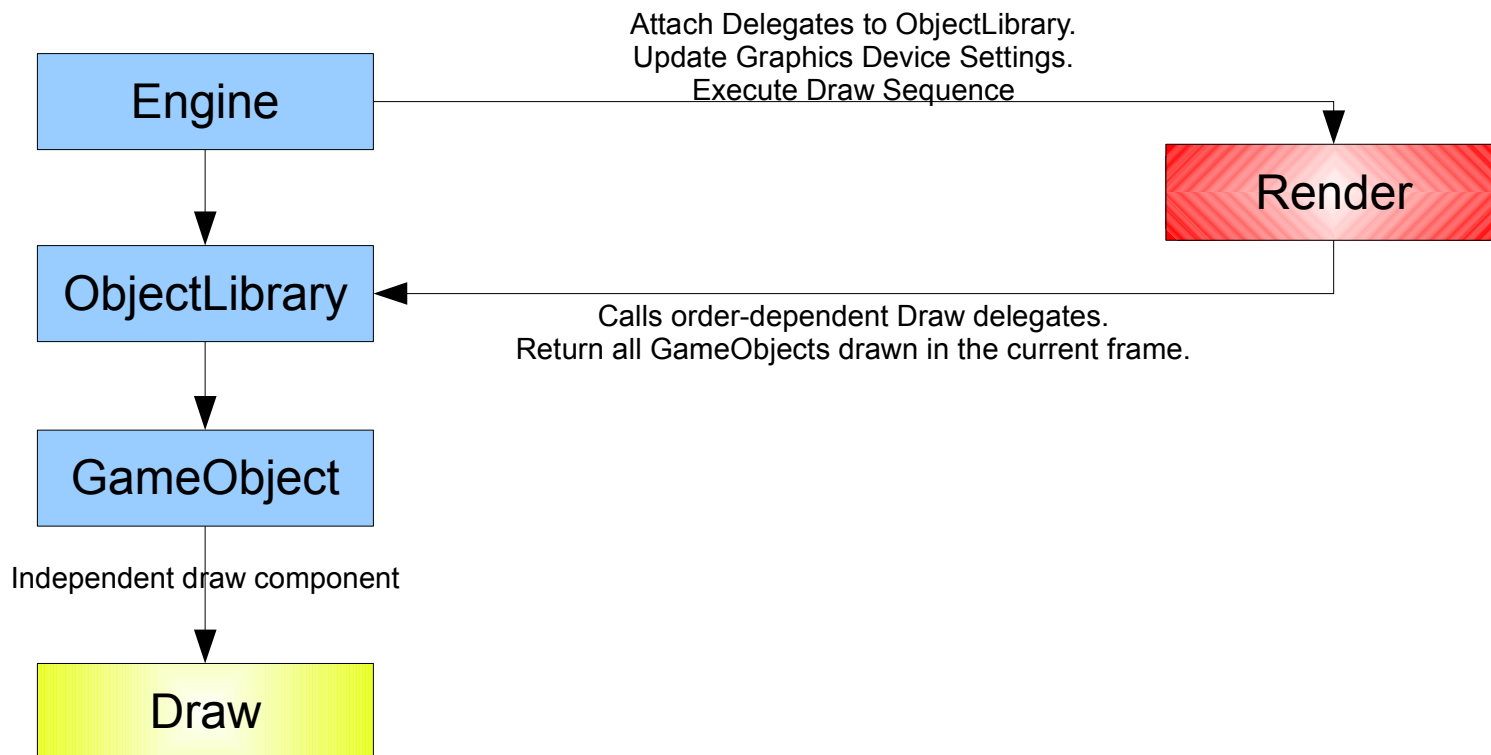
Scripted events attach to player through object collision response functions (see EventSystem).



Renderer

Renderer manages graphics device setup and shutdown operations for Engine through delegates, allowing Engine to manage the internal draw order of the following order-dependent categories.

- Object Library** - GameObjects or 3D World-space objects
- Sprites** - Billboards or 2D World-space objects
- Heads Up Display** - 2D Screen-space objects

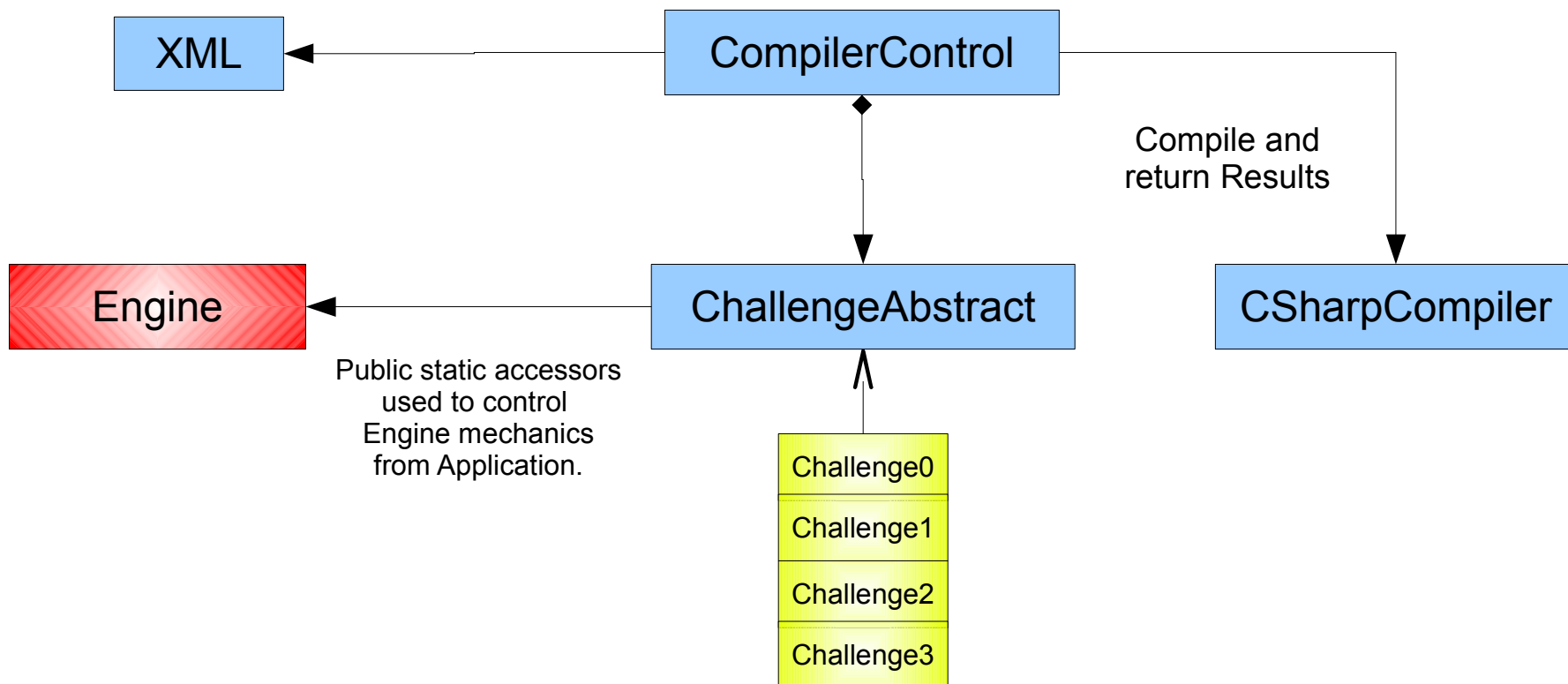


Compiler and Challenges

The CompilerControl loads the Challenges from XML. Compilation is triggered by user button-click, which sends the student code to the CSharpCompiler. The CSharpCompiler compiles the code in memory (though it can be compiled to an exe or a dll) and returns the CompilerResults to the CompilerControl,

Running the compiled code is also triggered by user button-click which calls the CS harpCompiler.execute() function, invoking the code in memory.

Aside from compilation, Challenges may verify student code through parsing (it will not run unless the code matches the parsing) allowing validation of code-style before triggering the first set of Events.



UCAD

