# OOP – Environment System

The goal of this lab is to study the **Environment System** and extend its functionality. You are **NOT allowed to directly edit** any of the contents of the **Core** and **Interface** namespaces (only the **Generator** namespace).

## Engine Modifications

We've seen that the engine is quite versatile and adding new objects with different behavior is fun. What's missing though?

### Step 1 – Validation

Maybe you haven't noticed, but the **Insert()** method in the Engine does not validate whether the object is **null**. However, we cannot simply edit the engine – we must obey the **open/closed principle** (open for extension, closed for modification) and avoid directly editing someone else's code. What we must do is **inherit the Engine class** and **extend** it using the **best OOP practices**.

Extend the engine and perform validation in the **Insert()** method.

**Note:** Do not repeat any of the base code! Find a way to reuse it.

### Step 2 – Pausing

Wouldn't it be fun if you could **pause** and **analyze the** **environment** at any given time with a **single press of a button**?

This is where the **IController** interface comes in. It defines **event** **Pause** and **void ProcessInput()**.

What is an event? An **event** is a special type that **keeps methods** and executes them when it (the event) 'happens'.

For example, we can **attach** a **Print()** method to a **MouseClick** event. When the **MouseClick** happens (when the mouse is clicked), it will execute our **Print()** method.

In our case, the **controller** will fire the **Pause** event whenever a certain button is pressed. Then, everything attached to the **Pause** event will be executed. What will we attach to the **Pause** event? – think about it.

* The new engine (the one that extends the old one) should accept an **IController** in its constructor. Create a **KeyboardController** that fires the **Pause** event whenever the **spacebar** is pressed, and pass it to the engine constructor.
* Modify your engine so that it pauses the game whenever the **Pause** event of the passed controller is fired (Hint: attach an event handler that pauses the engine when the event is fired).
* On each environment loop iteration, call the **ProcessInput()** method of the controller to check for any events.
* Again, **avoid repeating code** from the base engine implementation **at all cost**!