**What is the Xg 2D Pixel Library**

The Xg 2D Pixel Library is an object-oriented API used to create 2D game animations. Written as an extensive but compact package, it allows for the creation of sprites and the controls to modify their behavior. The Xg 2D Pixel Library defines sprints that can be viewed within a paper scene object. This object contains all the characters and animations that comprise a 2D game world. Characters are assigned behaviors with the creation of a finite-state-machine (FSM). This mechanism allows for a systematic approach to how sprites function and interact. As we will see through the following examples, developers can easily visualize their sprites behavior, design the behavior as a FSM and implement using the Xg 2D Pixel Library.

Xg is written in C++ and works seamlessly with OpenGl. Developers do not need an OpenGl background. This is a layer hidden from the developer and accessed only through the Xg 2D Pixel Library. This layer was added purposely in order to upgrade OpenGl later to other libraries, such as Vulkan.

The example code shown demonstrates a basic **XgPixel** main program. The creation of the **XgPixel** object is the controlling mechanism for creating, managing, and execution a Xg 2D Pixel Paper object. The **XgPaper** object defines all sprites, animations, FSMs, and camera positions. Without the paper object, there is nothing to animate.

The creation and execution of a Xg 2D Pixel game is a four-step process.

1. Create the **XgPixel** object. This is done through constructors that allow control of different overall aspects of the game. In this example, the constructor is defining the size of the animation window. But there are other constructors which will set the animation to full screen or determine the frames-per-second. This will be demonstrated in coming examples.
2. Create the **XgPaper** object. This example uses a paper object that has already been created. The details of this function are subject of coming sections. In short, this is where is all happens.
3. Add the paper object to the engine. After the paper object is created, it must be added to the engine. The engine can only execute one paper object at a time, adding an additional paper object will override the original.
4. Start the animation.