**2D Game Engine Development, Assignments 4 Question Sheet**

**Instructions:**

* In questions where you are asked to explain, please be concise.
* Show your work when necessary, be neat, precise, and brief!
* To help us grade your assignments and return this to you in a timely fashion please:
  + Put your name and answers in the answer sheet only (separate link provided). Anything you write outside of the answer sheet **will not** be graded.
  + Provide your answers in the order of the problems.
  + Please use only one side of 8.5x11 paper.
  + Please make sure you bring a hardcopy print out **of the answer sheet (!!only!!)** to submit at the beginning of class. Please do not print out the problems.

Your assignment may not be graded if any of the above is violated, you have been warned.

1. **Renderable objects:** as introduced in Example 3.1 (the Renderable Objects Project). If I create 3 instances of the ***Renderable*** objects for my game, how many copies of the GLSL vertex and fragment shaders will be created:
   1. 3 distinct copies of GLSL shaders will be created, one for each of the three ***SimpleShader*** objects to accompany the three ***Renderable*** objects.
   2. 3 distinct copies of vertex shaders will be created to share the same copy of fragment shader.
   3. One copy of vertex shader will be created to be shared among the three distinct copies of fragment shaders.
   4. One copy of vertex and fragment shaders will be created to be shared by all three instances of the ***Renderable*** objects.
   5. None of the above.
2. Continue with the three instances of ***Renderable*** objects from Question 1, in this case, how many instances of the ***Transform*** objects will be created?
   1. 3 distinct instances of the **Transform** objects will be created, one for each of the ***this.mXform*** instance variable of the three ***Renderable*** objects. As a result, the three ***Renderable*** objects can undergo different transformations and be drawn at independent places in the world.
   2. 1 instance of the **Transform** objects will be created to be shared by the ***this.mXform*** instance variable of the three ***Renderable*** objects. As a result, the three ***Renderable*** objects’ drawing transformations are relative to one another.
   3. 1 instance of the **Transform** objects will be created to be shared by the ***this.mXform*** instance variable of the three ***Renderable*** objects. The ***Transform.getXform()*** functionhas built-in intelligent support to allow the shared instance to support drawing the three ***Renderable*** objects at independent places in the world.
   4. 1 instance of the **Transform** objects will be created to be shared by the ***this.mXform*** instance variable of the three ***Renderable*** objects. It is the game developer’s responsibility to manage the shared **Transform** instance to ensure the three ***Renderable*** objects can be drawn to independent places in the world.
   5. None of the above
3. **True/False.** To accomplish similar results as in Example 3.2 (The Transform Objects Project), an alternative approach is to create a different WebGL vertex buffer for each Renderable object, transform the unit square vertices in JavaScript program, and send the transformed vertices to the vertex shader. In such case, the vertex shader program will have nothing to do but simply forwarding vertices to the ***gl\_Position***.
4. **Transformation:** The ***Transform*** utility class supports scaling (S), translation (T) and rotation (R) operations. Given a vertex V, the ***Transform.getXform()*** function returns a transform operator that performs which of the following transformation operation:
   1. Scaling first, followed by rotation, and lastly by translation, or T R S V. ***Note****,* V is the vertex, T R S V is Scaling *first*, *followed* by Rotation, and *lastly* Translation.
   2. T S R V
   3. S R T V
   4. S T R V
   5. R T S V
   6. R S T V
   7. All of the above are the same, so the answer is all of the above.
   8. None of the above is correct.
5. My *index.html* defines a 600x300 canvas area, my game defines the camera to be:

this.mCamera = new Camera(

vec2.fromValues(5, 10), // center of the WC

30, // width of WC

[0, 600, 0, 300] // viewport covers the entire canvas

);

this.mCamera.setBackgroundColor([1, 0, 0, 1]);

* 1. What is the height of my camera window?
  2. Given the viewport covers the entire canvas area, you know right in the middle of the viewport, the pixel coordinate is (300, 150). What is the world coordinate of this position?
  3. What is the world coordinate of the top-right corner of the viewport?
  4. You know the total area of the viewport is 600x300=180,000 pixel-square. What is the total area coverage of the camera window?

**2D Game Engine Development – Assignments 4 Answer Sheet**

**Name: Andy Tran**

**Score:**

|  |  |
| --- | --- |
| **Question #** | **Answer** |
| **1 (1pt)** | E |
| **2 (1pt)** | A |
| **3 (1pt)** | True |
| **4 (1pt)** | A |
| **5a (2pt)** |  |
| **5b (1pt)** |  |
| **5c (2pt)** |  |
| **5d (1pt)** |  |