CS460 Computer Graphics

Assignment 1: Basic Shapes Drawing. Total points 30
Deadline: 18 Oct, Monday

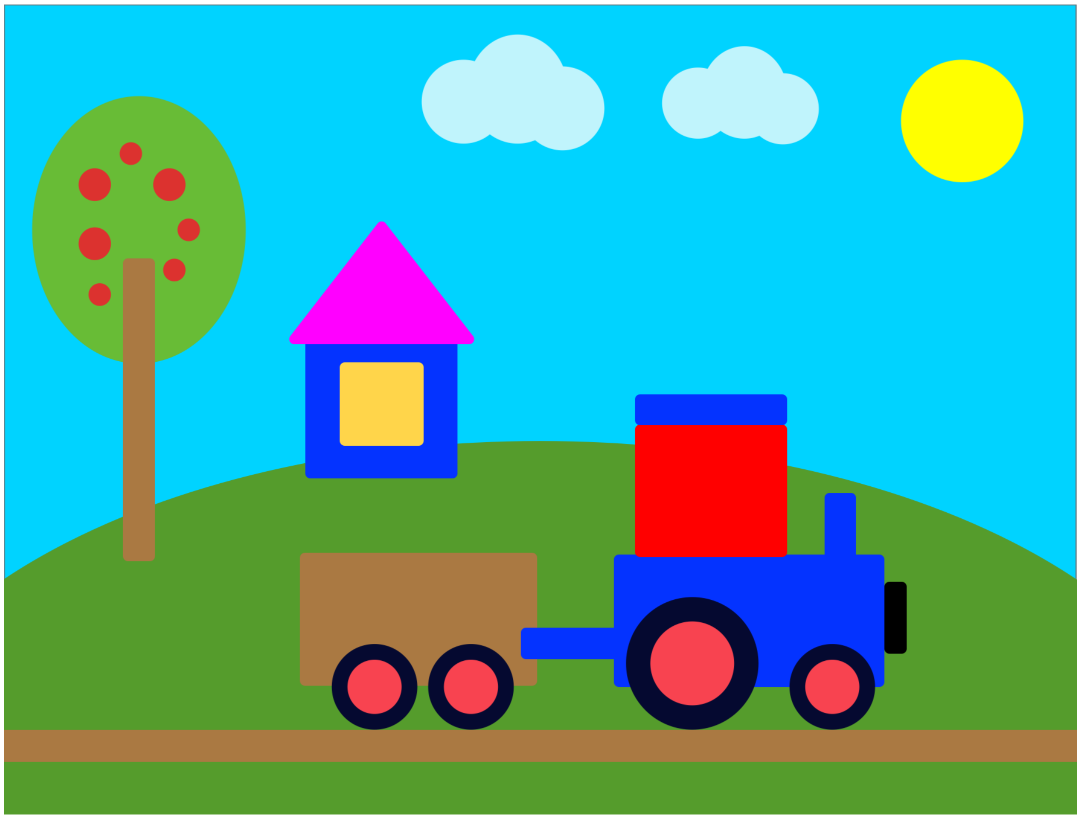
Description:

The purpose of this assignment is to practice transformation on various basic geometric shapes to create a dynamic drawing in an OpenGL canvas. The assignment assumes that you have studied the lectures and completed some of the example problems.

Instructions: Drawing a Landscape

The student can start with the drawing that they have implemented in the previous assignment. First, the student should make corrections to their assignment based on the feedback they received from their instructor. Specially, each drawing (home, tree, cloud, sun, train, wagon, background, etc) should have their own public classes. In addition, these classes should implement GShape interface and use the primitive classes like GCircle, GQuad, GOval, etc to create the drawing. The students must separate initialization from the rendering. Inside the render function, there must be no initialization.

In this assignment, the student are free to choose the dimension of the drawing. However, the drawing canvas must be bigger than 768 pixels width and 768 pixels height and less than 1920 pixels width and 1080 pixels height.



In this assignment only the landscape, train track, and the home can remain static. The rest of the drawing element must be transformed in a meaningful way to create animated dynamic environment. For example, the train moving along the track, clouds moving in the sky, sun glowing (as shown in the attached video), windmill rotating, etc. The wheels of the train should rotate as the train moves along the train track (as shown in the attached video).

Each drawing element must be element assuming it is drawn at the origin. Afterwards, the students can use any OpenGL translation functions to animate the drawing objects. When creating the scene, use the provided classes instead of starting from scratch. The students can use the provided set of classes from the previous assignment to begin working on their assignment.

In this assignment, each student is given option to create their own drawing objects in addition to the one from assignment one. In this scene, students must add 3 new drawing objects (not including the train, tree, cloud, and the sun drawing objects). Each drawing objects should be composed of a number of primitive drawing objects (Quad, Circle, Oval, Triangle, etc) and should be sufficiently complex. These drawing objects should be animated in the scene in a meaningful way. Examples, of the drawing objects that the students can consider can be, plane, helicopter, car, windmill, kite, UFO, rocket, robot, butterfly, ambulance, school-bus, firetruck, etc.

Req Type	**Marking**
R.x	Assignment gets 0 if any critical submission requirements (shown in red) are not met.
A.x	You lose 2 marks for each good practice requirements (shown in amber) not met.
G.x	You earn 2 marks for each design requirements (green) satisfied and well implemented; 1 mark if it's partly met or met but not well implemented; and 0 if it's not met.

Assignment Restrictions:

Technology Restrictions: You need to use the JOGL library to complete this assignment. You only need to submit the classes (java files) in an archive. You are free to use any IDE (Eclipse, etc) to complete the assignment.

The students must submit three screenshots or a short video (10 seconds) in order to demonstrate the animation in the drawing scene of their assignments.

Marking: This assignment is based on 4 design requirements numbered **G.1**...**G.4** for a total of 30 points. Points are awarded, or deducted, based on itemized requirements as follows:

Submission and Good Programming Practice Requirements

The following requirements pertain to all your assignments regardless of what your application is supposed to do (i.e. regardless of the design requirements). These requirements are to ensure that your code is usable, readable, and maintainable.

R.0 UNIQUENESS REQUIREMENT: The solution and code you submit MUST be unique. That is, it cannot be a copy of, or be too similar to, someone else's code, or other code found elsewhere. You are, however, free to use any code posted on our course website as part of our assignment solution. [Assigment mark =0 if this requirement is not met.]

R.1 CODE SUBMISSION ORGANIZATION AND COMPILATION: You should submit all the code files and data files necessary to compile and run your app. The professor will execute your java files by setting up an eclipse project. If you compress your submission on Blackboard, you must use only .zip format (not .rar or .jar or others). However, students are allowed to write code on Windows, Linux, or MacOS. The code should be generic enough to be OS agnostic. Make sure that there is no compiler errors in the submitted project. [Assigment mark =0 if this requirement is not met.]

R.2 README FILE: Your submission MUST include a README.txt file telling the professor how to setup and run your app. The professor should not have to look into your code to figure out how to create the project by using the submitted Java files. Your README.txt MUST contain the following:

- Your name, student number and email address and if you are working with a partner then their name, student number and email address as well.
- Issues: List any issues that you want the instructor to be aware of when they are evaluating your submission. In particular, tell us what requirements you did not implement or that you know are not working correctly in the submitted code. Here you are giving us your own assessment of your submitted project. [Assigment mark =0 if this requirement is not met.]
- Version: JDK version, OS you tested on your code on.

A.3 VARIABLE AND FUNCTION NAMES: All of your variables and functions should have meaningful names that reflect their purpose. Don't follow the convention common in math courses where they say things like: "let x be the number of customers and let y be the number of products." Instead, call your variables **numberOfCustomers** or **numberOfProducts**. Your program should not have any variables called "x" unless there is a good reason for them to be called "x". (One exception: It's OK to call simple for-loop counters i, j and k etc. when the context is clear and VERY localized.) [Minus 5 marks from assignment if this requirement is not met.]

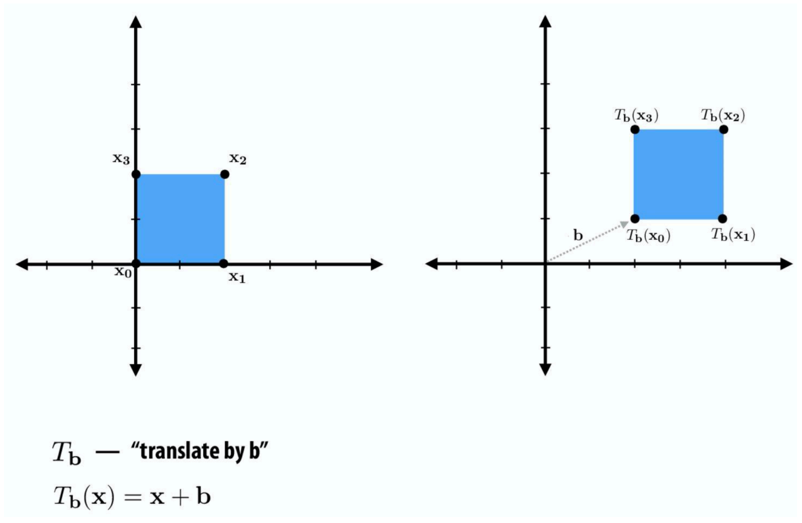
A.4 COMMENTS: Comments in your code must coincide with what the code actually does. It is a very common bug to modify code and forget to modify the comments and so you end up with comments that say one thing and code that actually does another. By the way, try not to over-comment your code but instead choose good variable names and function names that make the code more self-commenting. Don't be afraid to create local variables so that the variable name provides more clarity. [Minus 5 marks from assignment if this requirement is not met.]

A.5 CITATION REQUIREMENT: If you use code from other sources you should cite the source in comments that appear with the code. If the source is an Internet website then put the URL in the comments. You may use bits of code from outside sources but this may not form the complete solution you are handing in. You DON'T have to cite demo code we provide on the course web site or with tutorials and assignments, however that code should not be used for things you post publicly (like on GitHub). [Minus 5 marks from assignment if this requirement is not met.]

VERY IMPORTANT: Any sample code fragments provided may have bugs (although none are put intentionally). You must be prepared to find errors in the requirements and sample code. Please report errors so they can be fixed and an assignment revision posted.

Here are the core requirements of this assignment.

G.1: ORIGIN BASED DRAWING: All the drawing objects should be modified so that they are drawn with respect to the origin. For example in the figure below, the rectangle has been drawn with respect to the origin. We can then use transformation (including translation, rotation, and scaling) in order to position the drawing in the drawing canvas. [5 points]



G.2: FIVE NEW DRAWING ELEMENTS: Students must create 5 animated drawing objects. Of them there should be 3 new animated drawing objects. You can reuse 2 old drawing objects and animate them (you can reuse train, sun, or tree). Each drawing objects should be composed of a number of primitive drawing objects and involve sufficient complexity. These drawing objects should be animated in the scene in a meaningful way. Examples, of the new drawing objects that the students can consider can be, plane, helicopter, car, windmill, kite, UFO, rocket, robot, butterfly, ambulance, school-bus, firetruck, etc. [10 points]

G.3: DYNAMIC SCENE DRAWING: The initialization of the drawing objects must be separated from the rendering algorithm. The animation of the drawing objects should be meaningful. All animations must be carried out by using transformation (translate, scale, rotate, etc). The wheels of the train should be animated (as demonstrated in the attached video) as the train moves along the track, etc. Similar animation method would be encouraged for other drawing objects as well. In the assignment submission, three pictures or a video should be attached to demonstrate the animation of the drawing objects in the scene. [15 points]

[Bonus] G.4: LOADING DATA FROM FILE: In the project we create the drawing objects to draw a scene. At the initialization we use a number of data to initialize a drawing object.

In the bonus challenge, the students should create a text file and store these data in the file. The project should then read these data to initialize the drawing objects. A programmer/instructor should be able to change the data defined in the text file to set a different location for the drawing objects or different color for the drawing objects.

In order to accomplish this task, we need to define certain format and data requirement for the drawing objects. For example, we can consider the following format to store the drawing object data in the text file:

```
#GL_HOME
10 20 0 // location
1.0, 1.0, 0.0 // color for the top triangle
1.0, 0.0, 0.0 // color for the main quad
0.0, 1.0, 0.0 // color for the window

#GL_TREE
5 20 0 // location
1.0 1.2 1.0 // scaling
1.0, 0.0, 0.0 // color for the apple
0.0, 1.0, 0.0 // color for the leaves
1.0, 0.0, 1.0 // color for the trunk
```

While reading this text file, the program will find the number symbol followed by the drawing object type name. So, in the first case, the program is going to create a home object. For the first object, we are going to read the location and the color of the the triangle, quad, and windows. We need to specify comment for each values, however, the program is going to discard the comment part and skip to the next line to read the next entry from the file.

For each drawing objects, we may need different types of values and that is fine as long as we use comments to explain them. We also need to use these values to initialize the drawing object in the program. After creating each drawing objects, we can add them to the ArrayList and the ArrayList object will in turn display the drawing objects in the render function.

[5 points]

Please let me know if you have any questions. Thank you.