CSE102 – Computer Programming with C (Spring 2021)

Summer Project – 2D CAD Library

Handed out: Monday July 19, 2021.

Due: Start of CSE241 course in Fall 2021.

Hand-in Policy: Hand in directly to CSE241 course instructor (will let you know the submission details).
Collaboration Policy: No collaboration is permitted.
Grading: This project will contribute a small amount to your coursework for CSE241 in Fall 2021. The amount of the contribution is to be decided by the course instructor.

Description: You will implement a simple 2D CAD editor in C using Blend2D (https://blend2d.com/ or a similar vector graphics engine). You will also implement a C library to help create and manipulate 2D Computer Aided Design (CAD) content to be used by your editor. Examples for 2D CAD software include https://www.qcad.org/ and https://librecad.org/. The expected features of the editor and the library is provided below.

CAD2D Library

Your library will have the following functions. Note that there may be more (conceptually overloaded) instances of these functions. A minimal subset is provided below. Some functionality is left open for you to define and implement. A question mark (?) in the function name implies that you are expected to define appropriate name(s) for possible multiple instances of the function.

Note that a CAD object has many parts with multiple hierarchies. An example is given in Figure 1.

- **`CAD2D * c2d_start?():`**
  `CAD2D * c2d_start?(double width, double height):`
  `CAD2D * c2d_start?(double width, double height, Hierarchy * h):`
  Initializes your CAD content. A CAD content is initialized optionally on a canvas of a given dimension (`width` × `height`). Anything that you draw on the canvas should be within this limit. Note that the coordinate system of your canvas starts from the bottom-left corner (see Figure 2 for an illustration of the canvas geometry). x axis is towards left and y axis is perpendicular in the up direction. If no canvas is initialized, assume no bounds. Every CAD entity belongs to a hierarchy. The default hierarchy is the highest level. One can change or reassign hierarchies. Hierarchies can be nested.
- **`Label * c2d_add_point?(CAD2D * cad, double x, double y):`**
  `Label * c2d_add_point?(CAD2D * cad, Point2D p):`
  `Label * c2d_add_point?(CAD2D * cad, Point2D p, const Hierarchy * h):`
  `Label * c2d_add_point?(CAD2D * cad, Point2D p, const Hierarchy * h, const Label * l):`
  Add a point to the current or the given hierarchy. This function returns a pointer to the label attached to the created point unless it is given by the calling function. See the note in the data structures related to the uniqueness of the labels.
- **`Label * c2d_add_<BASIC>?(CAD2D * cad, ...):`**
  `Label * c2d_add_<BASIC>?(CAD2D * cad, ...):`
  `Label * c2d_add_<BASIC>?(CAD2D * cad, ..., const Hierarchy * h):`

`Label * c2d_add_<BASIC>?(CAD2D * cad, ..., const Hierarchy * h, const Label * l):`

Like point adding, add a new basic entity. Basic entities can be lines, arcs, circles, ellipses, splines, polylines, texts, polygons (including filled versions) and images. You will need appropriate data structures (as arguments defining their properties) to set these basic entities. Assume that any 2D basic shape (including points) can have color, thickness, and line style as well. You should enumerate these basic entities to cover the full requirement of the project (see the additional requirements).

- **`void c2d_snap(CAD2D * cad, const Label * ls, const Label * lt):`**

  Snaps a given entity (in ls) to another entry (lt). The following rules apply:

LS	LT	Snap Action
Point	Point	Snaps the point in LS to point in LT.
Point	Line (or Polyline)	Snaps the point in LS to the closest end of the line(s).
Point	Polygon	Snaps the point to the center of the polygon.
Point	Arc	Snaps the point to the center of the arc.
Polyline	Point	Define a strategy!
Polyline	Polyline	Define a strategy!
Polyline	Polygon	Define a strategy!
Polygon	Point	Define a strategy!
Polygon	Polyline	Define a strategy!
Arc	Point	Define a strategy!
Arc	Polyline	Define a strategy!

- **`double c2d_measure(CAD2D * cad, const Label * ls, const Label * lt):`**

  Measure the distance between two entities as defined in the following table:

LS	LT	Distance Definition
Point	Point	Euclidean distance between two points.
Point	Polyline	The distance closest line.
Point	Polygon	The distance to the closest edge of the polygon.
Point	Arc	The shortest distance to the arc.
Any Item	Any Item	The shortest distance between these 2D shapes.
Any Item	Any Item	The shortest distance between the centers of the 2D shapes. May need to define the center for each shape.

- **`Hierarchy * c2d_create_hierarchy?(CAD2D * cad):`**
  `Hierarchy * c2d_create_hierarchy?(CAD2D * cad, …):`
  `Hierarchy * c2d_create_hierarchy?(CAD2D * cad, Hierarchy * parent):`

Creates a hierarchy for a given CAD model. A new hierarchy can be at the highest level or a child of a parent. Note that a unique identifier for the hierarchy is needed unless the user provides it.

- **`void c2d_export(CAD2D * cad, char * file_name, char * options):`**

  Exports the current CAD to a file with the given options. The option indicates the targeted output file format. For now, consider:

  EPS: For the file format please refer to the attached document "Encapsulated PostScript File Format Specification Version 3.0.pdf".

  GTUCAD: Your own file format that can be imported.

- **`CAD2D * c2d_import(char * file_name, char * options):`**

  Import a CAD model from a given file. For now, consider only .GTUCAD files for which you will define a format.

Additional requirements:

1. You are supposed to have at least the following data structures:
   - **`CAD2D:`** The data structure to hold the current CAD content.
   - **`Point2D:`** A data structure to hold coordinates of 2D points (and maybe more).
   - **`Hierarchy:`** Holds the hierarchy information for CAD entities. These hierarchies can be nested or tree like.
   - **`Label:`** A label for a given CAD entity. Note that the label instances should be unique. Whenever you create a new label your library should make sure that it is not repeated.
2. Your library, at the minimum, should be able to generate the CAD examples shown in Figure 3 and Figure 4.
   - Create two .GTUCAD files implementing the CAD shown in in Figure 3 and Figure 4.
   - Your editor should be able to read these files and show the content properly.

CAD2D Editor

This editor should use CAD2D library as the base and deliver the following functionality to user of the editor. Note that there are a lot of design choices regarding the menus, content editing and visualization. You are expected to analyze a couple of simple 2D CAD editors and come up with an effective editor (this is rather a tall request but appropriate for an open-ended summer project).

1. Input and Output: The editor should load .GTU files. It should export .GTU as well as .EPS files (see the CAD2D library description for input and output files).
2. Viewing: The 2D CAD content should be shown to the user along with the canvas if any.
   a. Zoom: The visualizer should allow zooming in an out. You can limit the zoom-out to full canvas size. You can also limit the zoom-in to %1 of the canvas size.
   b. Picking: The user should be able to pick a CAD entity. You should allow the user to pick lowest hierarchy item closest to the user click location.
   c. Hierarchy navigation: User should be able to navigate the hierarchy of a given item. For example, if an item picked at the lowest hierarchy, user could go up the hierarchy visualizing the items (e.g., use + button).
3. Editing (optional). Although we call this an editor, editing features will be optional (and hopefully extra credit) as it involves a lot of additional design and programming.
   a. Line: Let the user enter a line by picking a start and end location. Snapping to a grid could be a useful idea.
   b. Rectangle: Let the user enter a rectangle like adding a line.
   c. Circle: Let the user enter a circle.

d. Translation: Let the user translate the picked item by dragging on the canvas.
e. Rotation: Let the user rotate a selected item.

What to hand in: You are expected to hand in all you source code (library and test programs) along with your makefile in a ZIP, RAR or similarly archived filed named "**cse102summerproject_lastname_firstname_studentno.zip**". When the makefile is run, it should compile everything and produce a test program. The test program should illustrate all the above functionality.

Appendix. Figures.

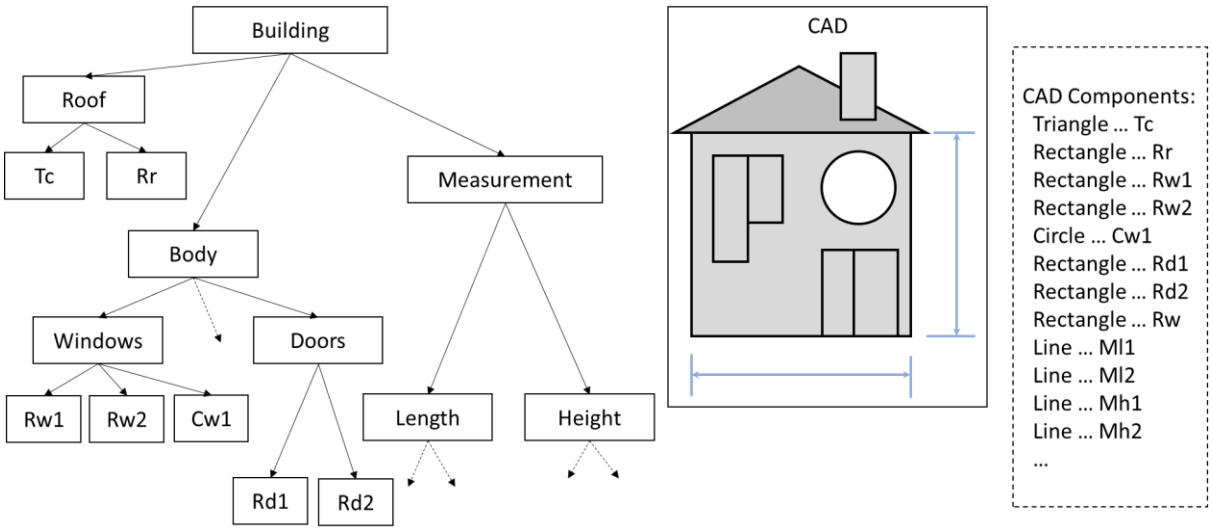

Figure 1 An example CAD (middle – a house) along with its hierarchies (left) and individual components (right).

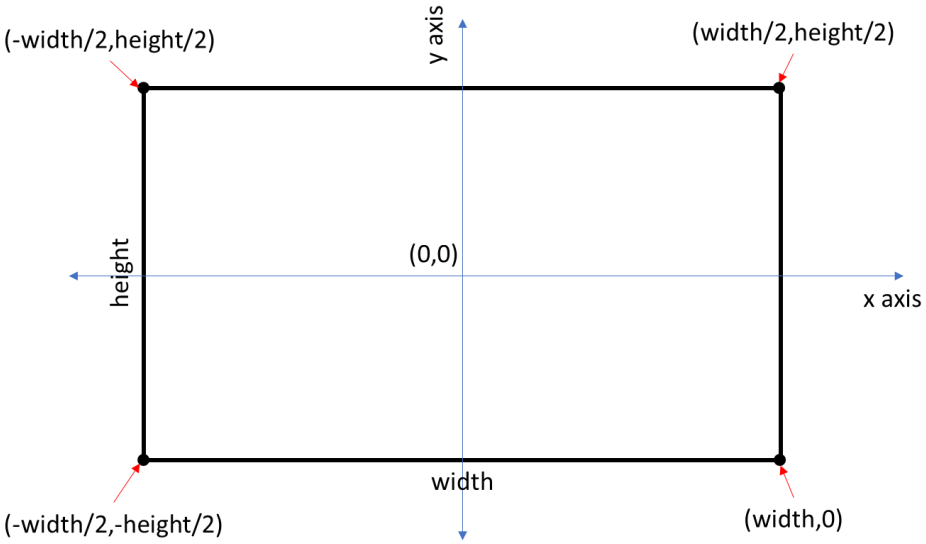

Figure 2 Canvas initialization for the CAD2D library with width and height and the assumed coordinate frame.

Figure 3 An example 2D CAD model visualized. Note that 2D entities have colors as well as other features. This figure does not show any hierarchy information.

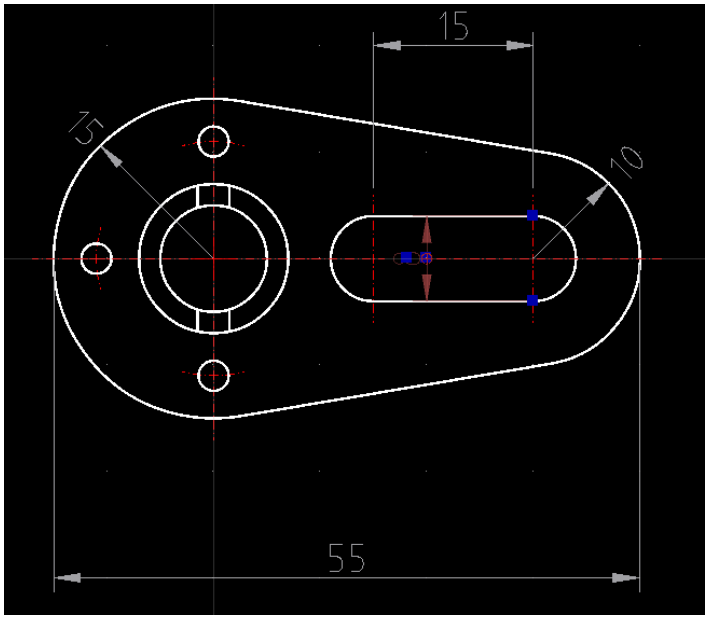

Figure 4 Another example 2D CAD. Note that text and numbers can be generated using basic entities (or alternatively by a defining a font – not required as part of this project).