[CPSC-4050/6050, SPRING 2020] ASSIGNMENT # 02: BEND IT LIKE BECKHAM!

THIS IS A ONE WEEK ASSIGNMENT WITH SUBMISSION DATE AS 18^{sth} FEB, 2020.

The objective of this assignment is to practice concepts behind curve drawing algorithms and put your knowledge of conics and splines to use. Apart from implementation of traditional mid-point algorithms for drawing curves such as circles and ellipses, you will exercise some your abilities to formulate, design algorithms and write code to draw a variety of free-form, piecewise curves maneuvered through the use of control points. Some of the free-form curves (splines) will interpolate through most of the control points (like cardinal splines) or approximate them such as with Bezier curves.

Objectives:

- Implement mid-point algorithm to draw a circle/ellipse and/or circular arcs.
- Develop an approach to rasterize parabolas and/or a high-order polynomial-based parameterized curve.
- Implement an algorithm for drawing Bezier curves of given order.
- [OPTIONALLY] Try out plotting splines (Cardinal or B-Splines).

Tasks and weightage in NUTSHELL:

BONUS scores will be clamped at 20%.

CPSC4050	CPSC6050
A. (30%) Implement mid-point algorithm for drawing circles	**A. (25%)** Implement any algorithm for drawing circular arcs passing through given three points. The first and the last points are the endpoints of the arc. [HINT] Perpendicular bisectors of the chords for a circle pass through the center for the circle. Intersection of such two perpendicular bisectors can be used to determine the circle center.
B. (40%) Implement a function to draw a parabola defined by the equation: $$(y - y_v)^2 = a(x - x_v)$$ where 'a' is an arbitrary constant to be computed from given inputs. (x_v, y_v) is the vertex point for the parabola and provided directly in the inputs.	**B. (25%)** Implement midpoint algorithm to draw a ellipse defined by the equation: $$\frac{(x - x_c)^2}{{r_x}^2} + \frac{(y - y_c)^2}{{r_y}^2} = 1$$ where (x_c, y_c) is the center for the ellipse, 'r_x' and 'r_y' are the radii along the X and Y axes respectively.

[CPSC-4050/6050, SPRING 2020] ASSIGNMENT # 02: BEND IT LIKE BECKHAM!

	(x_c, y_c) is provided directly in the inputs, 'r_x' and 'r_y' should be calculated from input data.
C. (30%) Draw a cubic Bezier curve. Inputs include the control points for the Bezier curve.	**C. (25%)** A sine wave can be expressed with its Taylor series expansion as: $$\sin x = \sum_{n=1}^{\infty} \frac{(-1)^{(n-1)} x^{(2n-1)}}{(2n-1)!}$$ Develop and implement an algorithm to plot half-a-cycle of a sine wave using polynomial parameterization of infinite order. Two points are provided in input. Difference between their X-coordinates equals half of the fundamental period for the sine wave. One of the input points is a crest (topmost point in the wave) and the other one is a trough (lowermost point in the wave). [HINT] To terminate the summation of the infinite, convergent series with an iterative approach, we can devise a termination criteria based on the magnitude of the change brought upon by each additional term in the series.
BONUS: (20%) implement midpoint algorithm for drawing ellipses as explained below, in Part B for 6050	**D. (25%)** Draw a quintic Bezier curve. Inputs include the control points for the Bezier curve.
BONUS: (20%) implement an algorithm for drawing cubic cardinal splines (Catmull-Rom splines) for a given set of control points. Refer to class videos, slides and/or Section 15.5.3. in the class textbook Fundamental of Computer Graphics 4th edition by Marschner and Shirley for details. Note that for Catmull-Rom splines, tension factor equals zero.	**BONUS: (20%)** implement an algorithm for drawing B-splines of order 3 (cubic). Refer to class videos, slides and/or Section 15.6.2. in the class textbook Fundamental of Computer Graphics 4th edition by Marschner and Shirley for details.

[CPSC-4050/6050, SPRING 2020] ASSIGNMENT # 02: BEND IT LIKE BECKHAM!

BONUS: (20%) Develop and implement midpoint algorithm for drawing a parabola defined by the equation: $$(y - y_v)^2 = a(x - x_v)$$ where 'a' is an arbitrary constant to be computed from given inputs. (x_v, y_v) is the vertex point for the parabola and provided directly in the inputs.	**BONUS: (20%)** implement an algorithm for drawing cubic cardinal splines (Catmull-Rom splines) for a given set of control points. Refer to class videos, slides and/or Section 15.5.3. in the class textbook Fundamental of Computer Graphics 4th edition by Marschner and Shirley for details. Note that for Catmull-Rom splines, tension factor equals zero.
	BONUS: (20%) Develop and implement midpoint algorithm for drawing a parabola defined by the equation: $$(y - y_v)^2 = a(x - x_v)$$ where 'a' is an arbitrary constant to be computed from given inputs. (x_v, y_v) is the vertex point for the parabola and provided directly in the inputs.

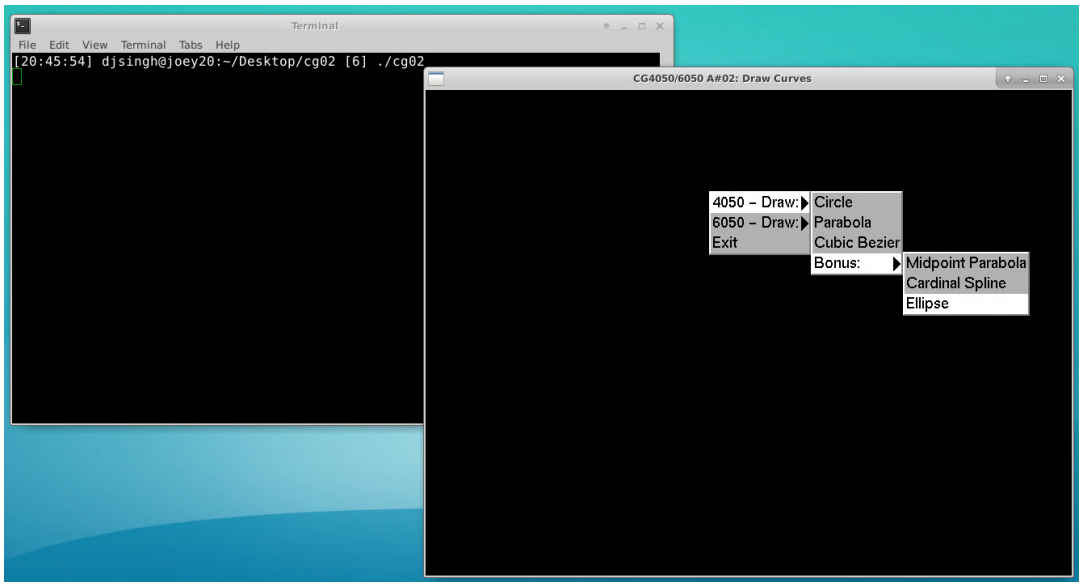
[CPSC-4050/6050, SPRING 2020] ASSIGNMENT # 02: BEND IT LIKE BECKHAM!

OS required: Linux, Compiler: C/C++, Build procedure: using make file. You are expected to work on general linux machines available for students @ SoC.

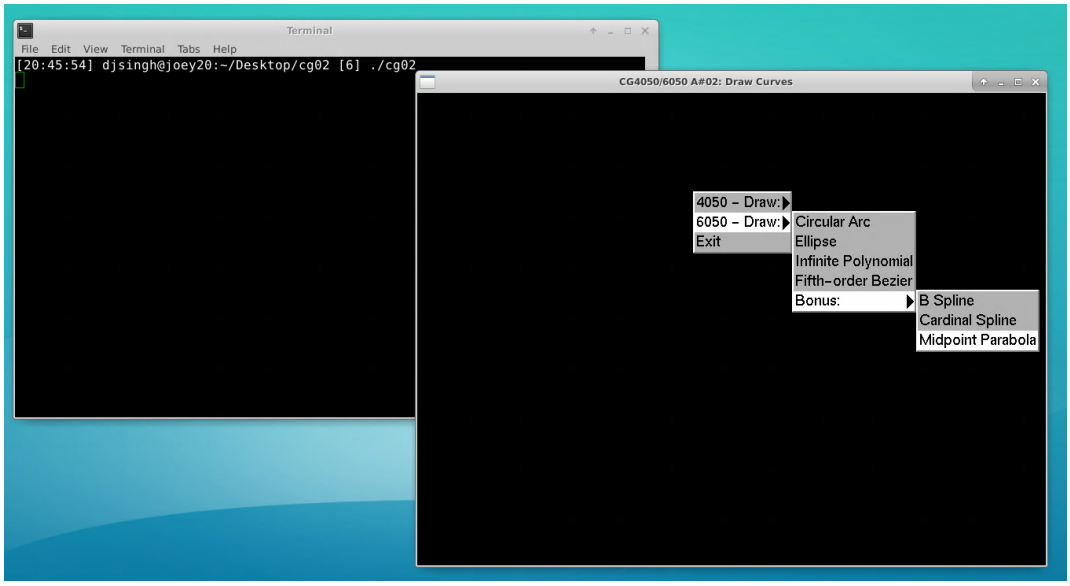
WorkFlow:

- Download the compressed package from https://clemson.instructure.com/courses/95982/files?preview=6108120 and save it in your local folder
- Open a terminal/command prompt and go to the local folder where your package was saved.
- Untar it with the command **tar –xzvf cg02.tar.** This will create a folder names **cg02**. Change your directory to this folder in your command prompt.
- This folder contains two source files main.cpp and stub.cpp and a Makefile file
- To build given package simply use the **make** command and it will produce an executable named **cg02.** Make sure you are in the same folder as the source code and the Makefile file. Now simply type the following at the prompt

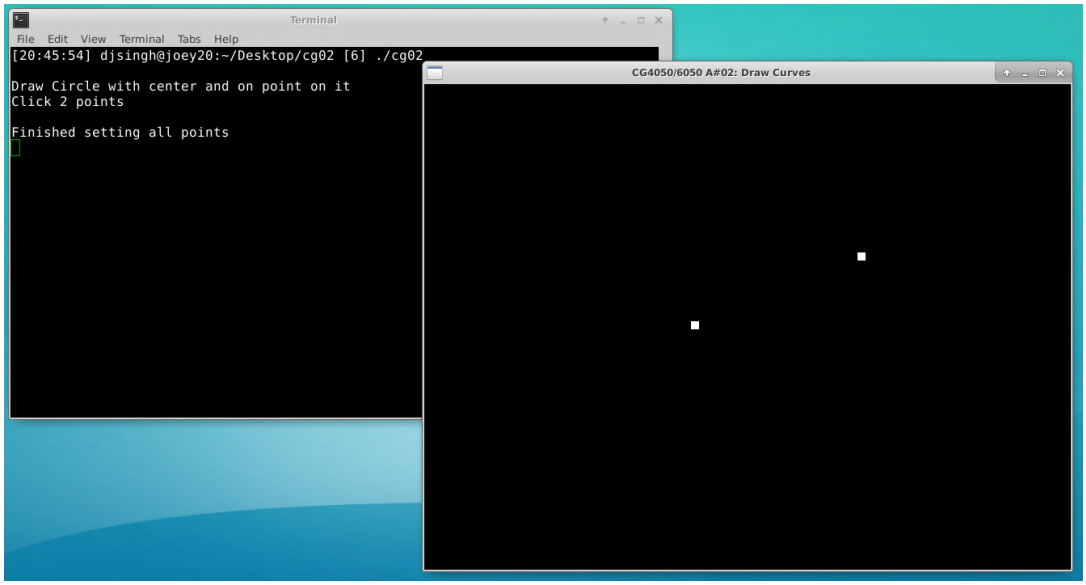
  >> make
- There are no elaborate make options in this package. So, if you want to make a clean build, you will have to manually delete the .o files and the executable file **cg02**
- To run the default program type **./cg02** at the prompt and it will open an OpenGL window with 800x600 pixels. The default coordinate system for this 2D window has its origin at the lowest-leftmost corner for the window and positive X-axis points from left to right while the positive Y-axis points from bottom to top. Right-click anywhere on this window and a context menu will pop-out. Few screenshots of this context menu are shown below.



[CPSC-4050/6050, SPRING 2020] ASSIGNMENT # 02: BEND IT LIKE BECKHAM!



- Clicking on one of the options from this context menu will print an informative message about how to provide necessary information (point locations) and the application will be in a recording mode until sufficient number of points have been provided through mouse clicks. When required number of points have been clicked a message stating “Finished setting all points” will be printed over the prompt and the application will be in some ‘drawMode’. Below is the example for drawing circles:



Note that two square dots highlight the points that were clicked.

- When the application enters ‘drawMode’, a callback function (written by you) corresponding to the drawing option selected from the context menu will be called by this application. In above case drawCircle() from stub.cpp will be invoked.

[CPSC-4050/6050, SPRING 2020] ASSIGNMENT # 02: BEND IT LIKE BECKHAM!

- You are only required to modify **stub.cpp** and provide implementation for respective functions in that file. You will only need to call **drawPixel()** from the rest of the package for this assignment and there is no need to write or use any OpenGL specific code explicitly. **You should not use any other OpenGL routine in you implementation and not change main.cpp**.
- Following is the mapping between given tasks and the functions in stub.cpp. Write your own code within designated functions and recompile all the files using 'make' command again. For a clean build, delete .o and cg02 files manually. **Make sure to backup your stub.cpp file periodically and not delete it accidentally during the cleanup.**

For CPSC4050 students

Task	Function
A. (30%) Implement mid-point algorithm for drawing circles	**drawCircle()** DETAILS: void drawCircle(*int centerX,* *int centerY,* *int pointOnCricleX,* *int pointOnCricleY*); Point (***centerX, centerY***) defines the center for your circle and, Point (***pointOnCricleX, pointOnCricleY***) lies on circle.
B. (40%) Implement a function to draw a parabola defined by the equation: $(y - y_v)^2 = a(x - x_v)$ where 'a' is an arbitrary constant to be computed from given inputs. (x_v, y_v) is the vertex point for the parabola and provided directly in the inputs.	**drawParabola()** DETAILS: void drawParabola(*int vertexX,* *int vertexY,* *int pointOnParabolaX,* *int pointOnParabolaY*); Point (***vertexX, vertexY***) defines the parabola vertex (x_v, y_v), and Point (***pointOnParabolaX, pointOnParabolaY***) lies on parabola.

[CPSC-4050/6050, SPRING 2020] ASSIGNMENT # 02: BEND IT LIKE BECKHAM!

C. (30%) Draw a cubic Bezier curve. Inputs include the control points for the Bezier curve.	**drawCubicBezier()** DETAILS: void drawCubicBezier(int* ptX, int* ptY); **ptX** is the pointer to an array of X-coordinates for the control points. **ptY** is the pointer to an array of Y-coordinates for the control points. **(ptX[n], ptY[n])** is the n^{th} control point starting with n = 0. For cubic Beziers n <=3.
BONUS: (20%) implement midpoint algorithm for drawing ellipses as explained below, in Part B for 6050	**drawEllipse()** Refer to Part B for 6050 below for details.
BONUS: (20%) implement an algorithm for drawing cubic cardinal splines (Catmull-Rom splines) for a given set of control points. Refer to class videos, slides and/or Section 15.5.3. in the class textbook Fundamental of Computer Graphics 4^{th} edition by Marschner and Shirley for details. Note that for Catmull-Rom splines, tension factor equals zero.	**drawCardinalSpline()** DETAILS: void drawCardinalSpline(int* ptX, int* ptY, int controlPointCount); **ptX** is the pointer to an array of X-coordinates for the control points. **ptY** is the pointer to an array of Y-coordinates for the control points. **controlPointCount** is the number of control points.

[CPSC-4050/6050, SPRING 2020] ASSIGNMENT # 02: BEND IT LIKE BECKHAM!

	(ptX[n], ptY[n]) is the nth control point. For cubic cardinal splines n < controlPoint. Count starting with n=0.
BONUS: (20%) Develop and implement midpoint algorithm for drawing a parabola defined by the equation: $$(y - y_v)^2 = a(x - x_v)$$ where 'a' is an arbitrary constant to be computed from given inputs. (x_v, y_v) is the vertex point for the parabola and provided directly in the inputs.	**drawMidpointParabola()** DETAILS: void drawMidpointParabola(int vertexX, int vertexY, int pointOnParabolaX, int pointOnParabolaY); Point (***vertexX, vertexY***) defines the parabola vertex (x_v, y_v), and Point (***pointOnParabolaX, pointOnParabolaY***) lies on parabola.

For CPSC6050 students

Task	Function
A. (25%) Implement any algorithm for drawing circular arcs passing through given three points. The first and the last points are the endpoints of the arc. [HINT] Perpendicular bisectors of the chords for a circle pass through the center for the circle. Intersection of such two perpendicular bisectors can be used to determine the circle center.	**drawArc()** DETAILS: void drawArc(int ptX1, int ptY1, int ptX2, int ptY2, int ptX3, int ptY3); Point (***ptX1, ptY1***) is the start point for the arc. Point (***ptX2, ptY2***) lies on the arc. Point (***ptX3, ptY3***) is the end point for the arc.
B. (25%) Implement midpoint algorithm to draw a ellipse defined by the equation:	**drawEllipse()** DETAILS:

[CPSC-4050/6050, SPRING 2020] ASSIGNMENT # 02: BEND IT LIKE BECKHAM!

$$\frac{(x-x_c)^2}{{r_x}^2}+\frac{(y-y_c)^2}{{r_y}^2}=1$$ where (x_c, y_c) is the center for the ellipse, 'r_x' and 'r_y' are the radii along the X and Y axes respectively. (x_c, y_c) is provided directly in the inputs, 'r_x' and 'r_y' should be calculated from input data.	void drawEllipse(int centerX, int centerY, int ptX1, int ptY1, int ptX2, int ptY2); Point (***centerX, centerY***) defines the center (x_c, y_c) for the ellipse, and points (***ptX1, ptY1***) and (***ptX2, ptY2***) lie on given ellipse.
C. (25%) A sine wave can be expressed with its Taylor series expansion as: $$\sin x = \sum_{n=1}^{\infty} \frac{(-1)^{(n-1)} x^{(2n-1)}}{(2n-1)!}$$ Develop and implement an algorithm to plot half-a-cycle of a sine wave using polynomial parameterization of infinite order. Two points are provided in input. Difference between their X-coordinates equals half of the fundamental period for the sine wave. One of the input points is a crest (topmost point in the wave) and the other one is a trough (lowermost point in the wave). [HINT] To terminate the summation of the infinite, convergent series with an iterative approach, we can devise a termination criteria based on the magnitude of the change brought upon by each additional term in the series.	**drawPoly ()** DETAILS: void drawPoly(int ptX1, int ptY1, int ptX2, int ptY2); **(ptX1, ptY1)** and **(ptX2, ptY2)** are two end-points for the parametric sinusoidal curve.
D. (25%) Draw a quintic Bezier curve. Inputs include the control points for the Bezier curve.	**drawQuinticBezier()** DETAILS: void drawQuinticBezier (int* ptX,

	int* ptY); **ptX** is the pointer to an array of X-coordinates for the control points. **ptY** is the pointer to an array of Y-coordinates for the control points. **(ptX[n], pt[n])** is the n^{th} control point starting with n = 0. For cubic Beziers n <=5.
BONUS: (20%) implement an algorithm for drawing B-splines of order 3 (cubic). Refer to class videos, slides and/or Section 15.6.2. in the class textbook Fundamental of Computer Graphics 4^{th} edition by Marschner and Shirley for details.	**drawCubicBSpline ()** DISPLAY: void drawCubicBSpline(int* ptX, int* ptY, int controlPointCount); **ptX** is the pointer to an array of X-coordinates for the control points. **ptY** is the pointer to an array of Y-coordinates for the control points. **controlPointCount** is the number of control points. **(ptX[n], pt[n])** is the n^{th} control point. For cubic cardinal splines n < controlPoint. Count starting with n=0.
BONUS: (20%) implement an algorithm for drawing cubic cardinal splines (Catmull-Rom splines) for a given set of control points. Refer to class videos, slides and/or Section 15.5.3. in the class textbook Fundamental of Computer Graphics 4^{th} edition by Marschner and Shirley for details. Note that for Catmull-Rom splines, tension factor equals zero.	**drawCardinalSpline()** DETAILS: void drawCardinalSpline(int* ptX, int* ptY, int controlPointCount); **ptX** is the pointer to an array of X-coordinates for the control points.

[CPSC-4050/6050, SPRING 2020] ASSIGNMENT # 02: BEND IT LIKE BECKHAM!

	ptY is the pointer to an array of Y-coordinates for the control points. **controlPointCount** is the number of control points. **(ptX[n], pt[n])** is the n^{th} control point. For cubic cardinal splines n < controlPoint. Count starting with n=0.
BONUS: (20%) Develop and implement midpoint algorithm for drawing a parabola defined by the equation: $$(y - y_v)^2 = a(x - x_v)$$ where 'a' is an arbitrary constant to be computed from given inputs. (x_v, y_v) is the vertex point for the parabola and provided directly in the inputs.	**drawMidpointParabola()** DETAILS: void drawMidpointParabola(int vertexX, int vertexY, int pointOnParabolaX, int pointOnParabolaY); Point (***vertexX, vertexY***) defines the parabola vertex (x_v, y_v), and Point (***pointOnParabolaX, pointOnParabolaY***) lies on parabola.

- All input variables for all functions above are named in a self-explanatory manner.
- The prompt level messages are also self-explanatory. Kindly examine them well before starting your work.
- For all drawing tasks in the 'drawMode', control points can be dragged around interactively by keeping the left-button for the mouse pressed on a particular control point.
- For tasks such as drawCardinalSpline() where the number of control points are not predetermined, during the recording mode, you can press 'Escape' key to terminate recording of additional control points and this will automatically switch the application into some 'drawMode'.
- Provided code package is not meant to be most resilient to all sorts of interactions. If you face some difficulties then reinitialize the your drawMode through context menu or exit the application and restart **./cg02**.

[CPSC-4050/6050, SPRING 2020] ASSIGNMENT # 02: BEND IT LIKE BECKHAM!

- A note on drawPoly() and drawMidpointParabola() tasks: You need to document your derivations and formulations along with the algorithm (pseudo-code) for these tasks, in your report.
- Prepare a PDF report with screen-grabs for the output of your implementation for each of the task.
- Prepare and upload a single zip file as your submission package that should include:
  - The above report
  - All your source code files and the makefile with a readme.txt if the need be.

Reference material:

- Chapters 15 from the course book Fundamentals of Computer Graphics by Marschner and Shirley, 4th edition. You can access this book from O'Reilly's online resources.

SUBMISSION DEADLINE: 11:59 p.m., 18th February (Tuesday) 2020.

- A delay of upto 24 hours will result in a proportional reduction of scores by 10%.
- A delay of upto 48 hours will result in a proportional reduction of scores by 20%.
- Any further delay will result in a proportional reduction of scores by 40%.

If there is a delay of for any medical exigency or any other unavoidable exigency, kindly contact me to avoid score reductions.

Upload your package over CANVAS

TRIVIA: David Beckham is a former soccer captain for England who is most famous for his 'banana' kicks. 'Bend it like Beckham' is a UK movie about an aspirant girl who wants to make it big in the world of soccer and master banana kicks. These Banana kicks lend unusual bends to the soccer ball during its flight to trace out a hard-to-model, three-dimensional curve. The hope here is that you will manage to bend it like Beckham and master the goals for all curve drawing tasks put out in this assignment!