**AGGP 247**

**Lab 2**

**Part 1: Building More Tools**

**Reading:**

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/named-and-optional-arguments>

* Add the following methods to Grid2D
  + public void DrawLine(Line line, bool DrawOnGrid = true)
    - *DrawOnGrid parameter determines if you convert the locations from Grid space to Screen space. This is true by default*
    - *IF FALSE: Your data in your object is already in ScreenSpace*
  + public void DrawObject(DrawingObject lineObj, bool DrawOnGrid = true)
    - *DrawOnGrid parameter is used in the same way as DrawLine*
  + public float ScaleGrid2Screen(float value)

{

return (value \* GridSize);

}

* + public float ScaleScreen2Grid(float value)

{

return (value / GridSize);

}

* Create a new Script called DrawingTools
  + public static float V3ToAngle(Vector3 startPoint, Vector3 endPoint)
    - Use Atan2 to convert
    - don't forget to convert from radians
  + Public static float LineToAngle(Line line)
    - Calls V3toAngle using the information from the line object.
  + public static Vector3 RotatePoint(Vector3 Center, float angle, Vector3 pointIN)
    - For a given center point and angle, determines the new rotated of a given point (pointIN)

===============

If you need a review on rotating a point.

<https://www.khanacademy.org/partner-content/pixar/sets/rotation/a/rotation-lesson-brief>

xnew = point.X \* cos(angle) - point.Y \* sin(angle);

ynew = point.X \* sin(angle) + point.Y \* cos(angle);

===============

Part 2: Putting it work

* Replace the code to draw your origin with a Diamond DrawingObject
* Rotating Diamond
  + Rotate a dot around the origin of the grid
  + The size is 2 Grid units wide
  + Will be 7.5 units from the origin
  + Will rotate once every 5 seconds.
  + Depending on how you set this up, it will start at either (1,0,0) or (0,1,0)
* Hexagon
  + Use your RotatePoint Method to determine six points on a circle.
  + Use these points to create the six sides of a Hexagon.
  + Draw this Object and the points relative to the Grid space
  + Extra Credit:
    - Drawing this using LineStrip Drawmode.
* Line Object on the Grid
  + A folder of different Line Objects has been provided to you in lab 1
  + Pick Draw one of these objects and convert the class to the new DrawingObject class. Use the DrawObject Method of your Grid2D class to draw it on screen.
* Parabolas
  + Draw the following Parabolas on the grid.
    - y = x^2
    - y = x^2 + 2x+ 1
    - y = -2x^2 + 10x + 12
    - x = -y^3
  + Calculate the points on the parabolas on increments of 1 on the X Axis.
    - These points will be calculated at run time
  + (Do this on the Y axis for X = -y^3)
  + Extra Credit:
    - Have a toggle between 1 and .5 draw increments
      * (You can do this by creating two objects at runtime)

===============

Two forms of the Parabola, solved for Y

y = ax2+ bx + c

y = a(x – h)² + k

you can transpose X with Y to get a parabola that graphs vertically

===============

Extra Credit:

* Add support for your Rotation member in your DrawingObject
  + Demonstrate using the Diamond DrawingObject or another converted object from lab 1.
  + Object will rotate around it’s center (local (0,0) but will be drawn in position at location on the screen.

**Part 3: Interface**

* Setup Tab to toggle groups of objects to be draw.
  + Set 1: Objects
    - Rotating Diamond
    - Hexagon
    - Line Object
  + Each Parabola should be drawn by itself
  + Display a string on Screen indicating what you are drawing
* Set up ‘~’ to toggle showing\hiding application controls
  + Have your controls on a panel that you can hide\show