Game Engine Design

# Overview:

Bubble is a package of files that abstracts the window’s console api to use in game development. The library comes with a screen buffer that stores character/text data and their corresponding attribute values. The end user can draw primitive shapes such as; pixels, lines, triangles, quadrilaterals and circles in the command prompt window. Furthermore, quads can be used to; create, edit and store sprites. Bubble also comes with a simple 2D physics engine that can preform mathematical operations. The first version of Bubble will have all the listed features, in the future; networking and sound handling will be included.

# Features:

## Narrative:

* When the user calls the main entry point for Bubble, several subsystems will be initialized. These systems are the rendering log & error handling systems.

### Rendering:

* Shape:
  + For polygons, each shape can be expressed as cartesian coordinates. Therefore, the superclass of all polygons stores an array of coordinates. When a shape drawing routine is called, lines are drawn between each coordinate (loops back for 2D objects).
  + For circles, the center and radius is stored as its own object.

Sprites:

* + Sprites files are formatted as follows and should be loaded/saved in this manner:
    - <width><height><glyphs><attributes>
  + Sprite are rendered like quadrilaterals.
* All shapes and sprites are axis-aligned and inherits a vector point that sits in the center of the object.
* **struct** Vector\_t {
* COORD position;
* double angle;
* double magnitude;
* }
* **struct** Circle\_t {
* COORD center;
* int radius;
* }
* **struct** Shape {
* int ShapeType;
* union{
* COORD vertices[];
* Circle\_t circle;
* }
* wchar\_t glyph;
* WORD attribute;
* Vector\_t vector;
* };
* **struct** Sprite\_t {
* COORD top\_left\_corner;
* int width , height;
* wchar\_t glyphs[];
* WORD attributes[];
* Vector\_t vector;
* }

### Logging:

* When bubble is instantiated, open a file that can be written to. When a log event occurs, append to the file using this format:

<data> <time> | <process> | <tag> | <message>

* + Data: “yyyy-mm-dd”
  + Time: “hh:mm:ss”
  + Process: the name of the application.
  + Tag: one of the following:
    - Info
    - Error
  + Message: details of the event.
* When bubble is deconstructed (upon error or completed execution), the file should be flushed and closed.

### Error Handling:

* Using errno.h standard error numbers to define what type of error occurred.
* When a runtime error occurs before bubble can complete initialization. The error is written to stderr. However, when an error presents itself after initialization then an error message box will pop up with a short message stating that bubble has crashed then, wait for the user to press ok before writing the error to file and exiting.

## Physics:

* Line length:
  + The length between 2 points can be calculated using Pythagoras theorem.
* Dot product of vectors
  + The dot product of a unit vector can be found between 2 vectors.
* Simple motion.
  + A vector moving with distance and velocity.
* The game world is described in tiles. One character is equal to a tile. The screen is made of m x n characters where m is the width and n is height of the screen buffer. Tiles are made up of a x b pixels where a is the width and b is the height of the tile.

## Nomenclature:

1. Global variables:
   1. g\_<type\_abbr><variable\_name>
2. Local variables:
   1. <type\_abbr><variable\_name>
3. Function parameters:
   1. <variable\_name><type\_abbr>
4. Struct members
   1. m\_<type\_abbr><member\_name>