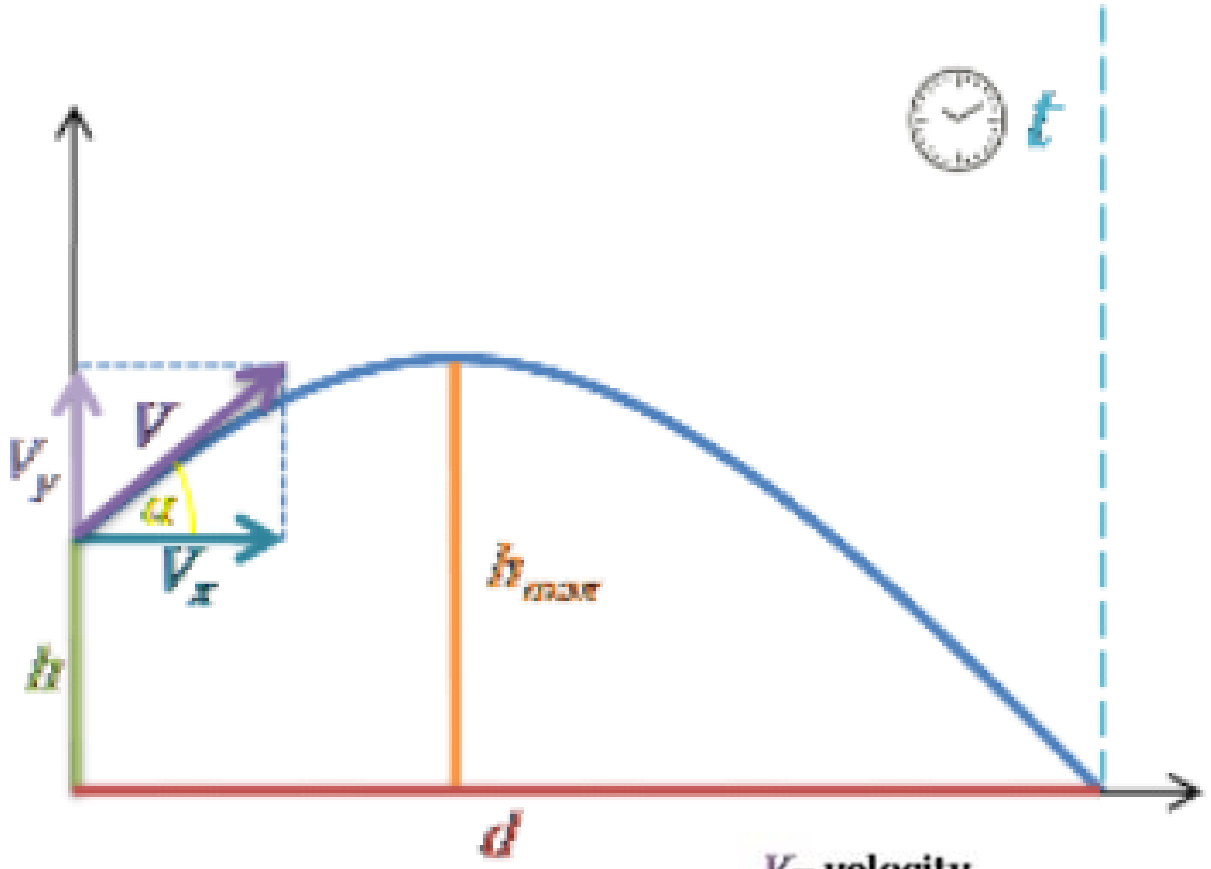
Projectile

Software Development for Robotics and Automation Engineering

Projectile

$$v_x = v \cdot \cos(\alpha)$$
$$v_y = v \cdot \sin(\alpha)$$

$$x = v_x \cdot t$$
$$y = h + v_y \cdot t - \frac{1}{2} g \cdot t^2$$



V – velocity
V_x – horizontal velocity
V_y – vertical velocity
α – angle of launch
h – initial height
t – time of flight
d – distance (range)
h_{max} – maximum height

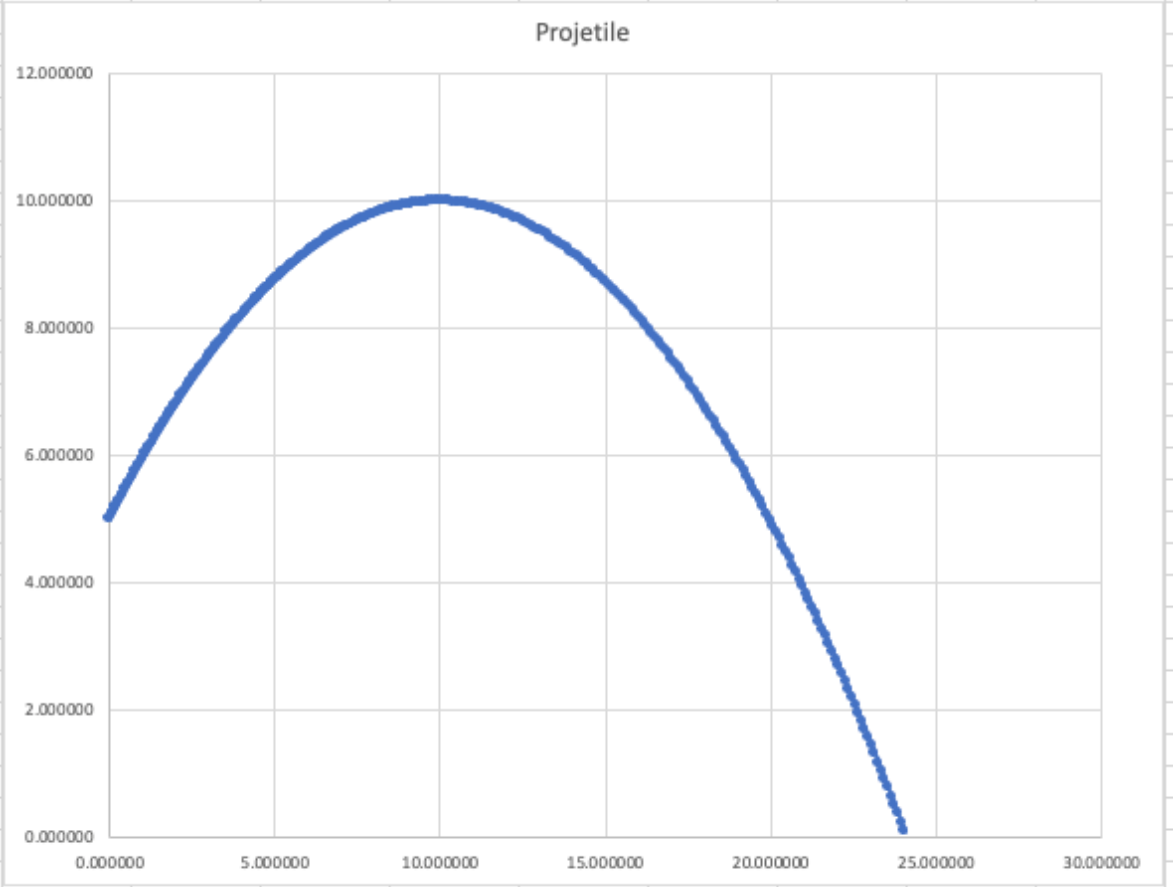
$$s = \sqrt{x^2 + y^2}\,, \qquad v = \sqrt{v_x^2 + v_y^2}\,, \qquad \alpha = \arctan(\frac{y}{x})$$

https://openstax.org/books/university-physics-volume-1/pages/4-3-projectile-motion

https://www.omnicalculator.com/physics/projectile-motion

Projectile with Excel

v0 [m/s]	14		
angle [degrees]	45		
height [m]	5		
time step [s]	0.01		
gravity [m2/s]	9.81		
	time	x	y
	0	0.000000	5.000000
	0.01	0.098995	5.098504
	0.02	0.197990	5.196028
	0.03	0.296985	5.292570
	0.04	0.395980	5.388132
	0.05	0.494975	5.482712
	0.06	0.593970	5.576312
	0.07	0.692965	5.668930
	0.08	0.791960	5.760568
	0.09	0.890955	5.851224
	0.1	0.989949	5.940899
	0.11	1.088944	6.029594
	0.12	1.187939	6.117307
	0.13	1.286934	6.204040
	0.14	1.385929	6.289791
	0.15	1.484924	6.374562
	0.16	1.583919	6.458351
	0.17	1.682914	6.541160
	0.18	1.781909	6.622987
	0.19	1.880904	6.703834
	0.2	1.979899	6.783699
	0.21	2.078894	6.862583
	0.22	2.177889	6.940487
	0.23	2.276884	7.017409
	0.24	2.375879	7.093351
	0.25	2.474874	7.168311
	0.26	2.573869	7.242291



Projectile with C

```
FRA142 — suriya@Suriyas-MBP —
time=2.269998, x=22.471796, y=2.196939
time=2.279998, x=22.570791, y=2.072758
time=2.289998, x=22.669785, y=1.947596
time=2.299998, x=22.768780, y=1.821451
time=2.309998, x=22.867775, y=1.694326
time=2.319998, x=22.966770, y=1.566221
time=2.329998, x=23.065763, y=1.437132
time=2.339998, x=23.164759, y=1.307066
time=2.349998, x=23.263754, y=1.176014
time=2.359998, x=23.362749, y=1.043985
time=2.369998, x=23.461742, y=0.910975
time=2.379998, x=23.560738, y=0.776981
time=2.389998, x=23.659733, y=0.642010
time=2.399998, x=23.758726, y=0.506056
time=2.409998, x=23.857721, y=0.369122
time=2.419998, x=23.956717, y=0.231205
time=2.429998, x=24.055712, y=0.092306
time=2.439998, x=24.154705, y=-0.047569
time=2.449998, x=24.253700, y=-0.188429
time=2.459998, x=24.352695, y=-0.330273
time=2.469998, x=24.451689, y=-0.473091
time=2.479998, x=24.550684, y=-0.616894
time=2.489998, x=24.649679, y=-0.761675
suriya:FRA142/ $
```

projectile.cpp > main(int, char * [])

```cpp
#include <stdio.h>
#include <math.h>

const float PI=3.1416;
const float Gravity=9.81;

int main(int argc, char *argv[])
{
    float x=0,y=0;
    float vx=0,vy=0;
    float time=0;

    float v0=14.0;
    float h=5.0;
    float alpha=45.0;
    float timestep=0.01;

    for(int i=0;i<250;i++)
    {
        x = v0*cos(alpha*PI/180)*time;
        y = h+v0*sin(alpha*PI/180)*time-Gravity*time*time/2;
        printf("time=%f, x=%f, y=%f\n",time,x,y);
        time=time+timestep;
    }

    return 0;
}
```

Projectile with C++

```
FRA142 — suriya@Suriyas-MBP — ..c
time=2.219998, x=21.976822, y=2.803137
time=2.229998, x=22.075817, y=2.683859
time=2.239998, x=22.174812, y=2.563601
time=2.249998, x=22.273808, y=2.442360
time=2.259998, x=22.372801, y=2.320143
time=2.269998, x=22.471796, y=2.196939
time=2.279998, x=22.570791, y=2.072758
time=2.289998, x=22.669785, y=1.947596
time=2.299998, x=22.768780, y=1.821451
time=2.309998, x=22.867775, y=1.694326
time=2.319998, x=22.966770, y=1.566221
time=2.329998, x=23.065763, y=1.437132
time=2.339998, x=23.164759, y=1.307066
time=2.349998, x=23.263754, y=1.176014
time=2.359998, x=23.362749, y=1.043985
time=2.369998, x=23.461742, y=0.910975
time=2.379998, x=23.560738, y=0.776981
time=2.389998, x=23.659733, y=0.642010
time=2.399998, x=23.758726, y=0.506056
time=2.409998, x=23.857721, y=0.369122
time=2.419998, x=23.956717, y=0.231205
time=2.429998, x=24.055712, y=0.092306
time=2.439998, x=24.154705, y=-0.047569
suriya:FRA142/ $
```

projectile2.cpp > main(int, char * [])

```
52  int main(int argc, char *argv[])
53  {
54      float v0=14.0;
55      float h=5.0;
56      float alpha=45.0;
57      float timestep=0.01;
58
59      vector2d init_pos(0.0,h);
60      vector2d init_vel(cos(alpha*PI/180.0)*v0, sin(alpha*PI/180.0)*v0);
61      vector2d init_accel(0.0,Gravity);
62      vector2d position;
63      float time=0.0;
64
65      while (position.getY()>=0)
66      {
67          position = init_pos + init_vel*time + init_accel*time*time*0.5;
68          printf("time=%f, x=%f, y=%f\n",time,position.getX(),position.getY());
69          time=time+timestep;
70      }
71
72      return 0;
73  }
```

SDL Projectile

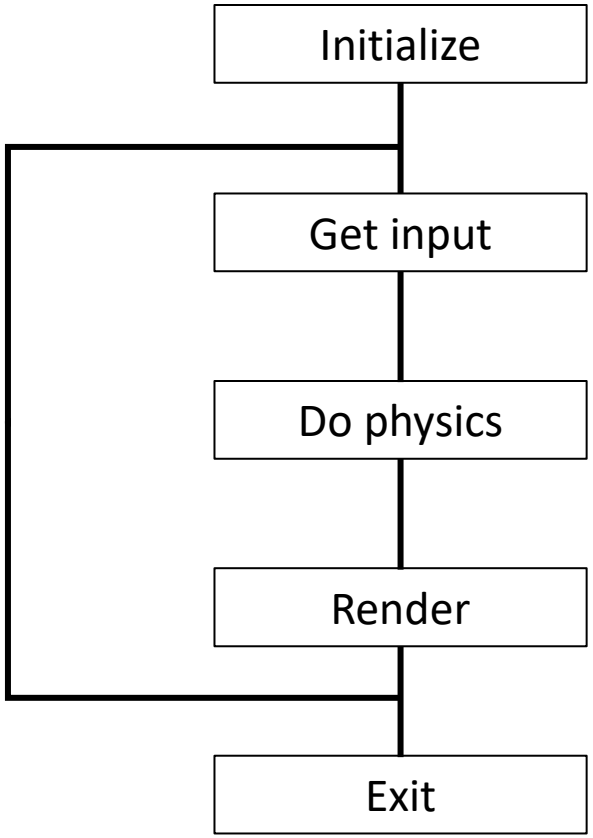
Software Development for Robotics and Automation Engineering

Vector2D Class

Vector2D
-m_x:float -m_y:float
+getX():float +getY():float +setX(float):void +setY(float):void +length():float +operator+(Vector2D):Vector2D +operator+=(Vector2D,Vector2D):Vector2D +operator-(Vector2D):Vector2D +operator-=(Vector2D,Vector2D):Vector2D +operator*(float):Vector2D +operator*=(float):Vector2D +operator/(float):Vector2D +operator/=(float):Vector2D +normalize()

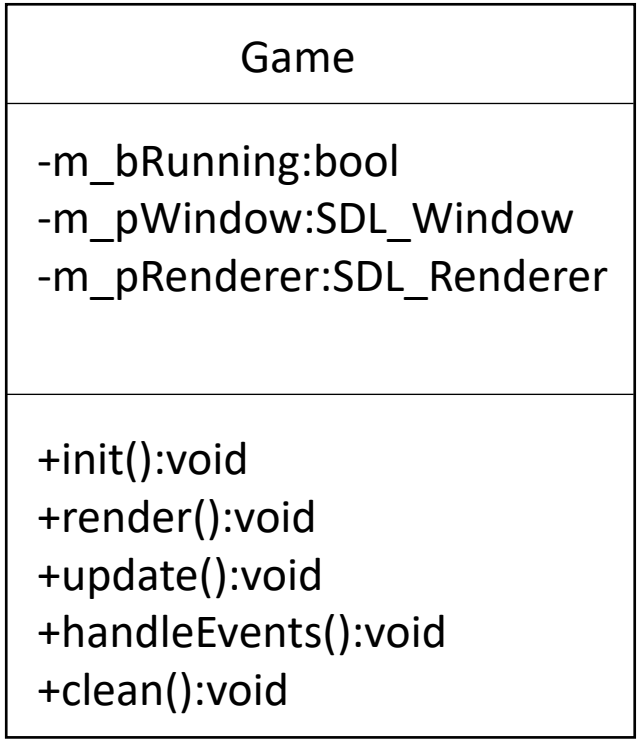
```
class Vector2D
{
public:
    float getX() {}
    void setX(float x) {}
    float lenth() {}
    Vector2D operator+(const Vector2D& v2) const {}
    Vector2D operator+=(Vector2D& v1,const Vector2D& v2) {}
    Vector2D operator-(const Vector2D& v2) const {}
    Vector2D operator-=(Vector2D& v1,const Vector2D& v2) {}
    Vector2D operator*(float scalar) {}
    Vector2D& operator*=(float scalar) {}
    Vector2D operator/(float scalar) {}
    Vector2D& operator/=(float scalar) {}
    void normalize() {}
protected:
    float m_x;
    float m_y;
};
```

Game Loop




```
void init(){}
void render(){}
void update(){}
void handleEvents(){}
void clean(){}

bool g_bRunning = true;
int main()
{
   init();
   while(g_bRunning)
   {
      handleEvents();
      update();
      render();
   }
   clean();
}
```

Game Class

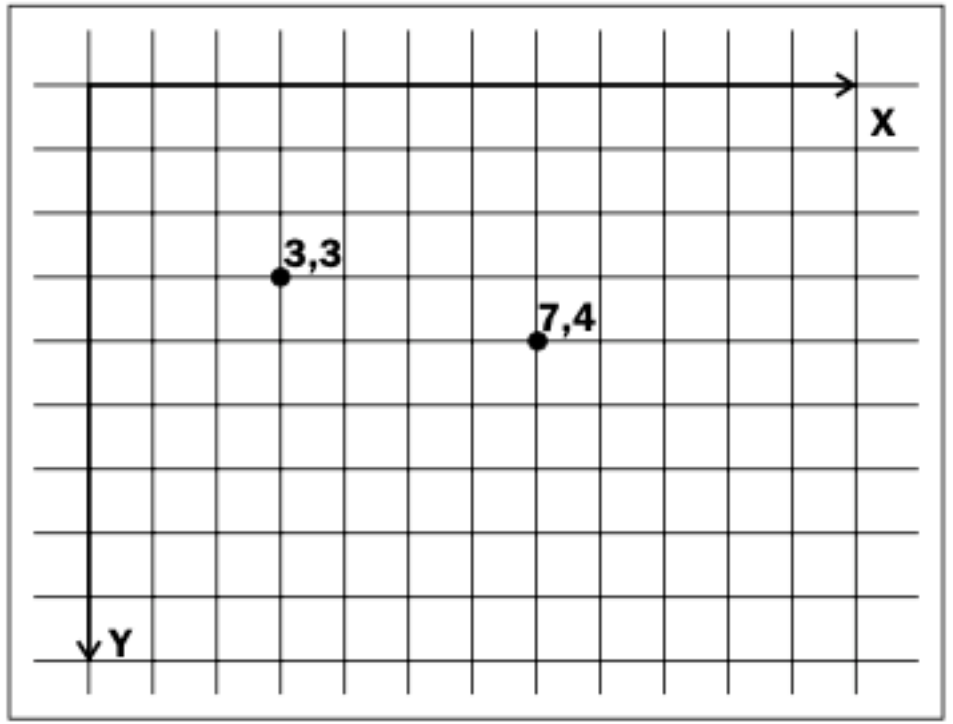



```
#ifndef __Game__
#define __Game__
#include “SDL.h”
class Game
{
    public:
        Game() {}
        ~Game() {}

        // simply set the running variable to true
        void init() { m_bRunning = true; }
        void render() {}
        void update(){}
        void handleEvents() {}
        void clean(){}

        // a function to access the private running variable
        bool running() { return m_bRunning; }
    private:
        bool m_bRunning;
        SDL_Window* m_pWindow;
        SDL_Renderer* m_pRenderer;
};

#endif /* defined(__Game__) */
```

Main Program

```
#include "Game.h”

// our Game object
Game* g_game = 0;

int main(int argc, char* argv[])
{
    g_game = new Game();
    g_game->init(”FRA142 Projectile", 100, 100, 640, 480, 0);
    while(g_game->running())
    {
        g_game->handleEvents();
        g_game->update();
        g_game->render();
    }
    g_game->clean();
    return 0;
}
```

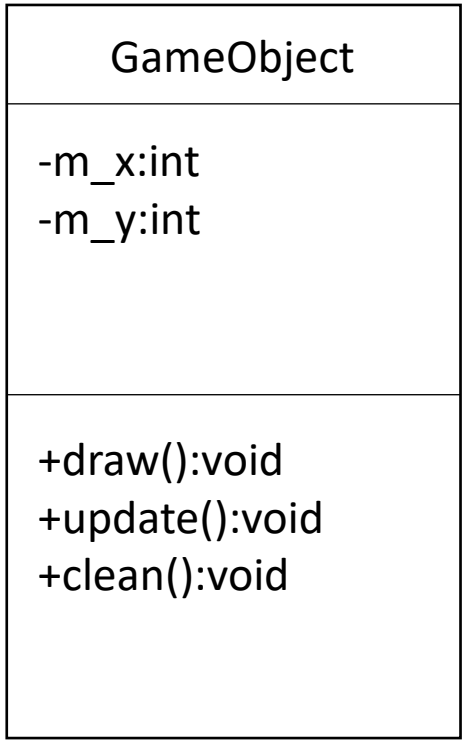
Drawing in SDL




```
// set to black
SDL_SetRenderDrawColor(g_pRenderer, 0, 0, 0, 255);
// clear the window to black
SDL_RenderClear(g_pRenderer);

//Set the color
SDL_SetRenderDrawColor(g_pRenderer, 255, 255, 255, SDL_ALPHA_OPAQUE);


SDL_RenderDrawLine(g_pRenderer, x1, y1, x2, x2);
SDL_RenderDrawPoint(g_pRenderer, x1, y1);


// show the window
SDL_RenderPresent(g_pRenderer);
```

https://www.widgetworx.com/projects/sl.html

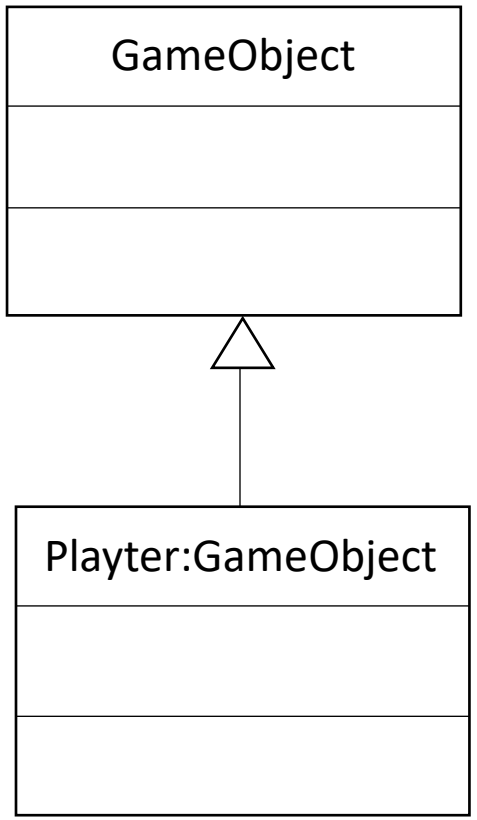
GameObject Class



- All games have objects, for example, players, enemies, non-player character (NPC), traps, bullets, and doors.
- These objects will have a lot of the same data and require a lot of the same basic functions.
- The use of inheritance, we can share common functionality between similar classes and also create subtypes from existing types.

```
class GameObject
{
public:
    void draw() { std::cout << "draw game object"; }
    void update() { std::cout << "update game object"; }
    void clean() { std::cout << "clean game object"; }
protected:
    int m_x;
    int m_y;
};
```

Player Class




```
class Player : public GameObject // inherit from GameObject
{
    public: void draw()
    {
        GameObject::draw();
        std::cout << "draw player";
    }
    void update()
    {
        std::cout << "update player";
        m_x = 10;
        m_y = 20;
    }
    void clean()
    {
        GameObject::clean();
        std::cout << "clean player";
    }
};
```

TextureManager Class

```
bool load(std::string fileName,std::string id, SDL_Renderer* pRenderer);

// draw
void draw(std::string id, int x, int y, int width, int height, SDL_Renderer* pRenderer, SDL_RendererFlip flip =
SDL_FLIP_NONE);

// drawframe
void drawFrame(std::string id, int x, int y, int width, int height, int currentRow, int currentFrame,
SDL_Renderer* pRenderer, SDL_RendererFlip flip = SDL_FLIP_NONE);
```

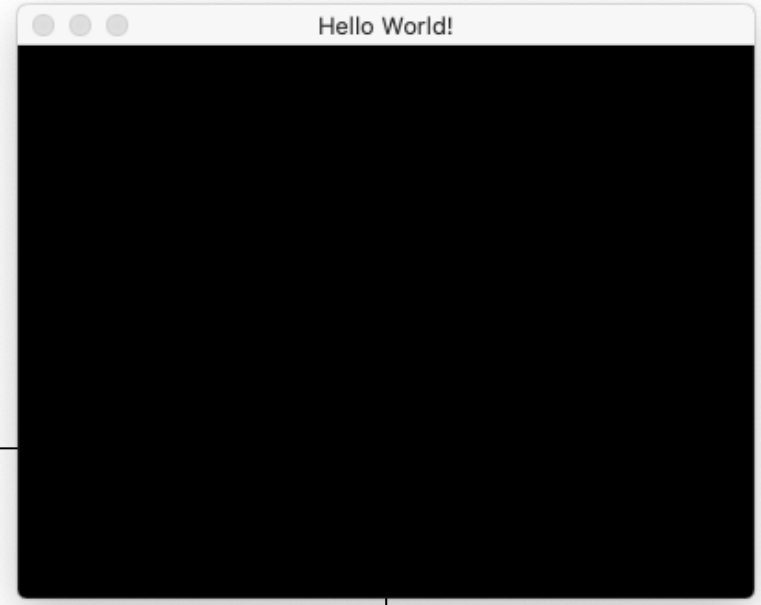
Projectile with PyGame

Software Development for Robotics and Automation Engineering

PyGame

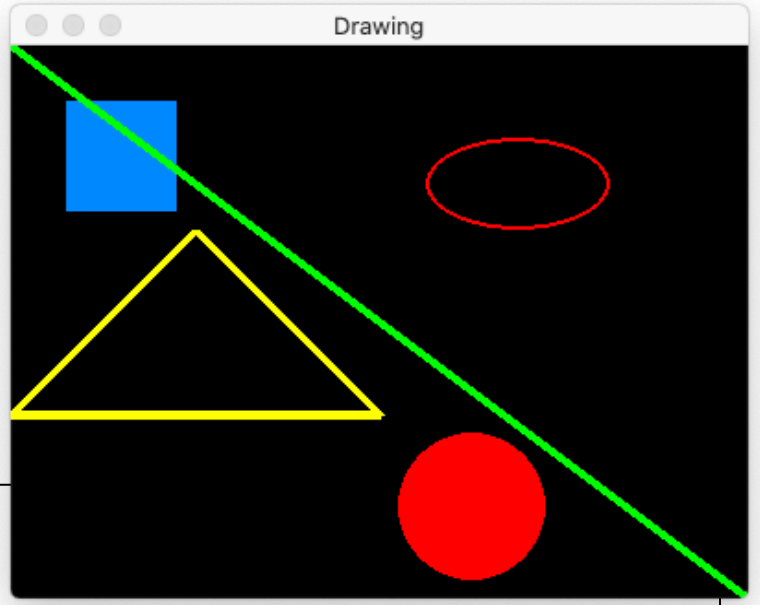
- pygame https://www.pygame.org/

Hello World




```
import pygame
import sys
from pygame.locals import *

pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Hello World!')

while True: # main game loop
        for event in pygame.event.get():
                if event.type == QUIT:
                        pygame.quit()
                        sys.exit()
        pygame.display.update()
```

Drawing

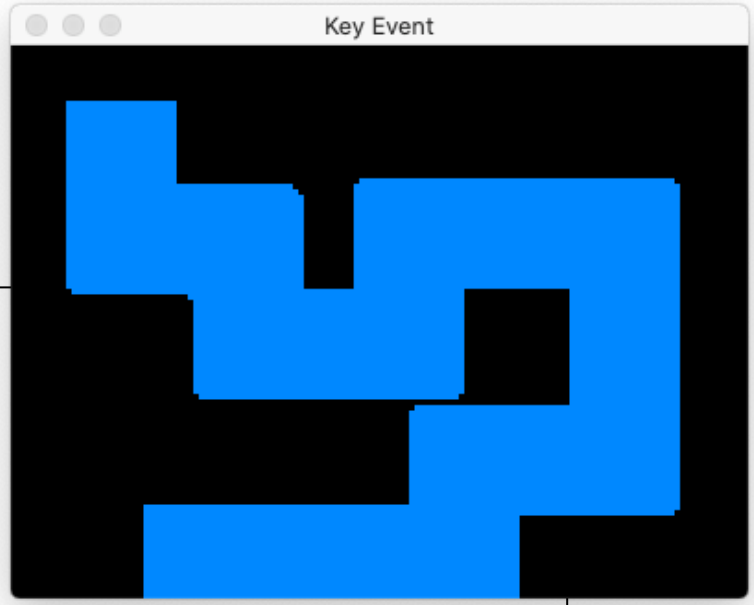



```
import pygame
import sys
from pygame.locals import *

pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Drawing')

while True: # main game loop
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
    pygame.draw.rect(DISPLAYSURF, (0, 125, 255), pygame.Rect(30, 30, 60, 60))
    pygame.draw.line(DISPLAYSURF, (0, 255, 0), [0, 0], [400, 300], 5)
    pygame.draw.ellipse(DISPLAYSURF, (255, 0, 0), [225, 50, 100, 50], 2)
    pygame.draw.circle(DISPLAYSURF, (255, 0, 0), [250, 250], 40)
    pygame.draw.polygon(DISPLAYSURF, (255, 255, 0), [[100, 100], [0, 200], [200, 200]], 5)
    pygame.display.update()
```

Text Rendering

```
import pygame
import sys
from pygame.locals import *
pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Text Rendering')

#load the fonts
font = pygame.font.SysFont("Times new Roman", 36)
# Render the text in new surface
text = font.render("FRA142 Projectile", True, (158, 16, 16))

while True: # main game loop
        for event in pygame.event.get():
                if event.type == QUIT:
                        pygame.quit()
                        sys.exit()
        DISPLAYSURF.fill((255,255,255))
        DISPLAYSURF.blit(text,(200 - text.get_width() // 2, 150 - text.get_height() // 2))
        pygame.display.update()
```

Key Event




```
import pygame
from pygame.locals import *

pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Key Event')
done = False
x = 30
y = 30

while not done: # main game loop
	for event in pygame.event.get():
		if event.type == QUIT:
			done = True
		pressed = pygame.key.get_pressed()
		if pressed[pygame.K_UP]: y -= 3
		if pressed[pygame.K_DOWN]: y += 3
		if pressed[pygame.K_LEFT]: x -= 3
		if pressed[pygame.K_RIGHT]: x += 3

	pygame.draw.rect(DISPLAYSURF, (0, 125, 255), pygame.Rect(x, y, 60, 60))
	pygame.display.update()
pygame.quit()
```

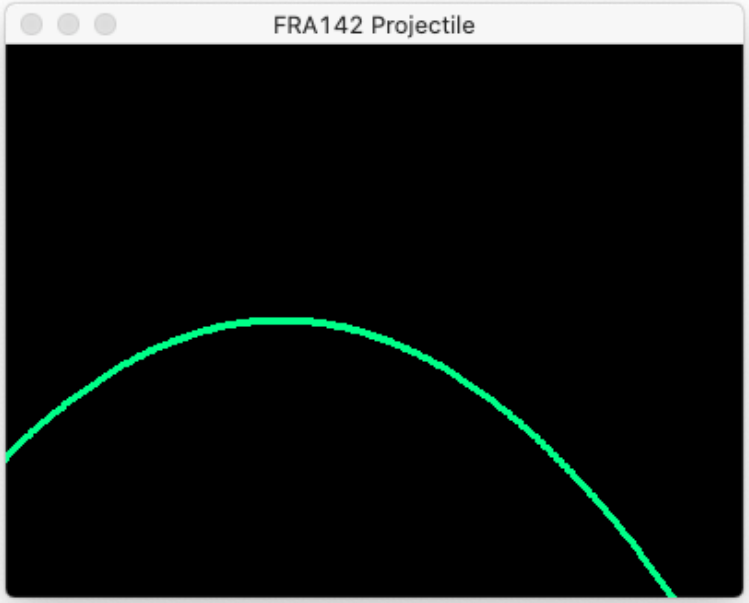
pygame.vector2

- Import pygame.math
- Numerical operations:

  vec+vec, vec-vec, vec*number, number*vec, vec/number, vec//number,

  vec+=vec, vec-=vec, vec*=number, vec/=number, vec//=number

Projectile

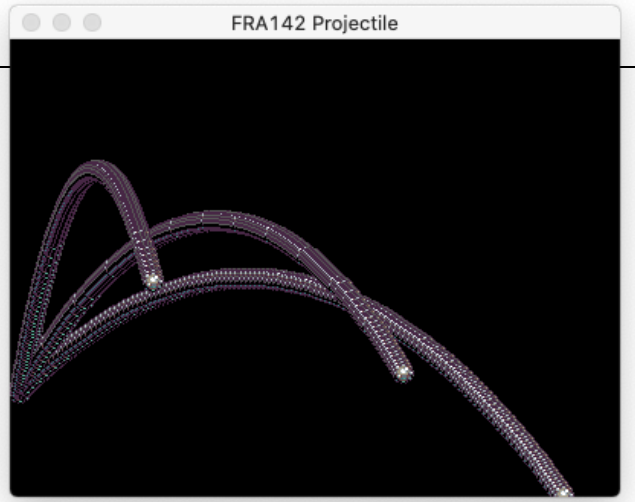



```
import pygame
import sys
from pygame.locals import *
import pygame.math
import math

pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('FRA142 Projectile')
done = False

v0 = 14
h = 5
alpha = 45
timestep = 0.01
gravity = -9.81
init_accel = pygame.Vector2(0,gravity)
init_vel = pygame.Vector2(math.cos(alpha*math.pi/180)*v0, math.sin(alpha*math.pi/180)*v0)
init_pos = pygame.Vector2(0,h)
pos = pygame.Vector2()
time = 0

while not done: # main game loop
    for event in pygame.event.get():
        if event.type == QUIT:
            done = True
            pressed = pygame.key.get_pressed()

        pos = init_pos + init_vel*time + init_accel*time*time*0.5
        time += timestep

        pygame.draw.circle(DISPLAYSURF, (0, 255, 125), [pos.x*15, 300-pos.y*15], 2)
        pygame.display.update()

pygame.quit()
```

Projectile OOP

projectile.py

```
import pygame
from pygame.locals import *
import pygame.math
import math
from bomb import Bomb

pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('FRA142 Projectile')
done = False
bomb1 = Bomb(0,5,14,45,0.01)
bomb2 = Bomb(0,5,14,60,0.01)
bomb3 = Bomb(0,5,14,80,0.01)
bomblist = [bomb1, bomb2, bomb3]
DISPLAYSURF.fill((0,0,0))

while not done: # main game loop
        for event in pygame.event.get():
                if event.type == QUIT:
                        done = True
                pressed = pygame.key.get_pressed()
        for i in range(len(bomblist)):
                bomblist[i].update()
                bomblist[i].draw(DISPLAYSURF)
        pygame.display.update()

pygame.quit()
```



bomb.py

```
import pygame
import pygame.math
import math
from pygame.locals import *

gravity = -9.81

class Bomb:
    def __init__(self, x, y, v0, alpha, timestep):
        self.image = pygame.image.load("assets/bomb.png")
        self.alpha = alpha
        self.pos = pygame.Vector2()
        self.init_pos = pygame.Vector2(x,y)
        self.init_vel = pygame.Vector2(math.cos(alpha*math.pi/180)*v0, math.sin(alpha*math.pi/180)*v0)
        self.init_accel = pygame.Vector2(0,gravity)
        self.time = 0
        self.timestep = timestep

    def draw(self, surface):
        #pygame.draw.circle(surface, (0, 255, 125), [self.pos.x*15, 300-self.pos.y*15], 2)
        surface.blit(self.image, [self.pos.x*15, 300-self.pos.y*15])

    def update(self):
        if self.pos.y >= 0:
            self.pos = self.init_pos + self.init_vel*self.time + self.init_accel*self.time*self.time*0.5
            self.time += self.timestep
```