HIT DETECTION PSEUDOCODE

// BALL CO-ORDINATES //

left_of_ball = ball->xPos()

right_of_ball = ball->xPos() + ball->width()

top_of_ball = ball->yPos()

bottom_of_ball = ball->yPos() + ball->height()

// PLAYER 1 CO-ORDINATES //

right_of_player_1 = player_1->xPos() + player_1->width()

top_of_player_1 = player_1->yPos()

bottom_of_player_1 = player_1->yPos() + player_1->height()

// PLAYER 2 CO-ORDINATES //

left_of_player_2 = player_2->xPos()

top_of_player_2 = player_2->yPos()

bottom_of_player_2 = player_2->yPos() + player_2->height()

// PLAYER 1 COLLISION //

if (left_of_ball < right_of_player_1 AND top_of_ball > top_of_player_1 AND bottom_of_ball < bottom_of_player_1)

THEN change x direction

// PLAYER 2 COLLISION //

if (right_of_ball > left_of_player_2 AND top_of_ball > top_of_player_2 AND bottom_of_ball < bottom_of_player_2)

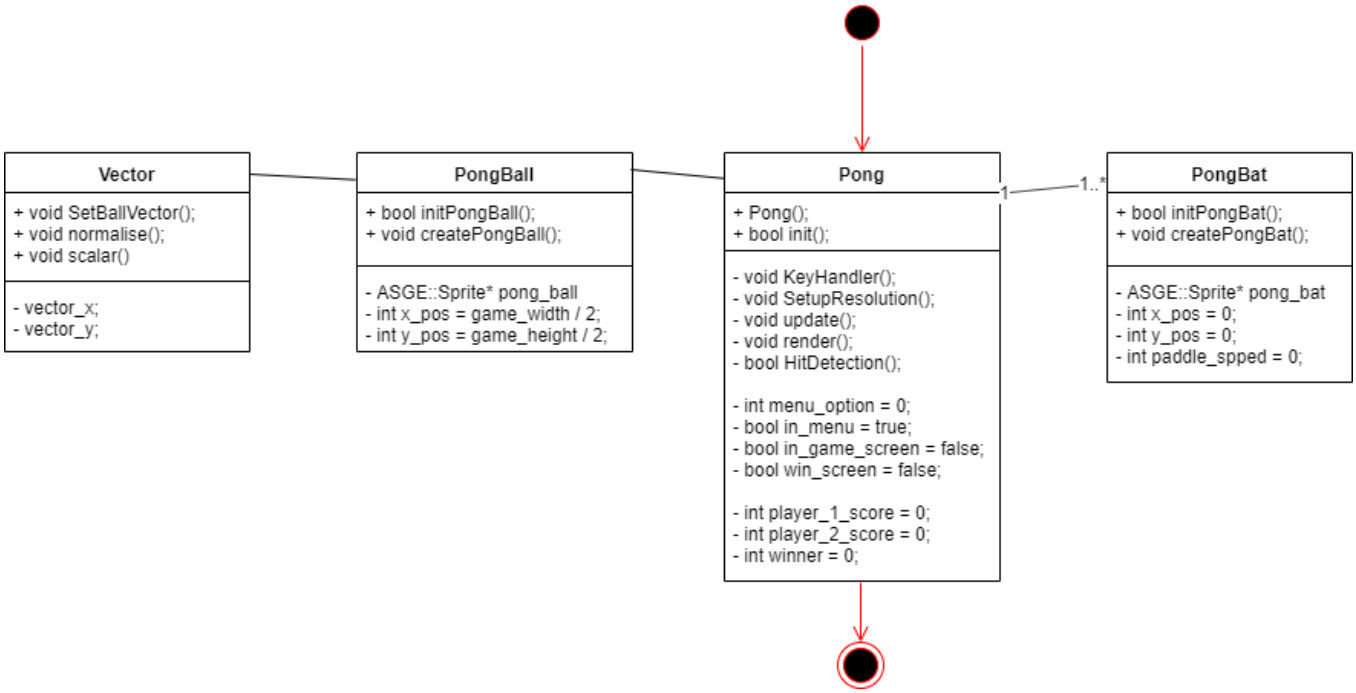
THEN change x direction

// SCREEN COLLISION //

if (top_of_ball <= 0 or bottom_of_ball >= bottom_of_screen)

THEN change y direction

UML DIAGRAM


Vector
+ void SetBallVector();
+ void normalise();
+ void scalar()
- vector_x;
- vector_y;
PongBall
+ bool initPongBall();
+ void createPongBall();
- ASGE::Sprite* pong_ball
- int x_pos = game_width / 2;
- int y_pos = game_height / 2;
Pong
+ Pong();
+ bool init();
- void KeyHandler();
- void SetupResolution();
- void update();
- void render();
- bool HitDetection();
- int menu_option = 0;
- bool in_menu = true;
- bool in_game_screen = false;
- bool win_screen = false;
- int player_1_score = 0;
- int player_2_score = 0;
- int winner = 0;
1
1..*
PongBat
+ bool initPongBat();
+ void createPongBat();
- ASGE::Sprite* pong_bat
- int x_pos = 0;
- int y_pos = 0;
- int paddle_spped = 0;