class EnemyOne : public Entity

{

private:

float oldX, oldY, oldAngle;

float rotation; // current rotation rate (radians/second)

VECTOR2 direction; // direction of rotation

float MapX;

float MapY;

float tempswaytime;

int swaycounter;

float range;

VECTOR2 center;

float angle;

bool near1;

vector<EnemyBullet\*> bulletlist;

int bulletno;

float dex;

float timerbullet;

float attack;

float spd;

int xp;

...

}

void EnemyOne::shootBullet(VECTOR2 player)

{

EnemyBullet \*a = new EnemyBullet(\*bTemp);

VECTOR2 start(MapX, MapY);

a->setMapX(start.x + WIDTH / 4);

a->setMapY(start.y + HEIGHT / 2);

player.x -= a->getWidth() / 2 \* a->getScale();

player.x -= WIDTH / 4;

player.y -= a->getHeight() / 2 \* a->getScale();

player.y -= HEIGHT / 2;

a->setvstart(start);

a->setvtarget(player);

bulletlist.push\_back(a);

}

void EnemyOne::drawBullet(float x, float y)

{

for (vector<EnemyBullet\*>::iterator it = bulletlist.begin(); it != bulletlist.end(); )

{

if ((\*it)->getdel() == true) //if bullets have collided with player, del of bullet set to true

{

SAFE\_DELETE( \* it);

it = bulletlist.erase(it);

}

else

{

++it;

}

}

for each(EnemyBullet\* p in bulletlist)

{

p->setY(p->getMapY() + y);

p->setX(p->getMapX() + x);

p->draw();

}

}

void Solitude::collisions()

{

VECTOR2 collisionVector;

// if collision between ship and planet

for (int a = 0; a < enemyoneList.size();a++)

{

for(int b = 0; b < enemyoneList[a]->getBulletlist().size();b++)

{

if (enemyoneList[a]->getBulletlist()[b] == NULL)

return;

if (enemyoneList[a]->getBulletlist()[b]->collidesWith(player, collisionVector))

{

player.damage(enemyoneList[a]->getAttack());

enemyoneList[a]->getBulletlist()[b]->setdel(true);

}

}

}

}

EnemyOne::~EnemyOne()

{

delete bTemp;

for (int i = 0; i < bulletlist.size(); i++)

SAFE\_DELETE(bulletlist[i]);

}