**Work in Progress Report**

**Papa John’s Kool Coding Klub**

Major developments/breakthroughs(reference specific code please):

* Player now faces the mouse cursor. This makes aiming bullets much easier
  + public void setAngle(Vector3 vMousePos) {

fAngle = MathUtils.*atan2*((vMousePos.y - getY()), (vMousePos.x - getX()));

}

* Bullets have been added to the game with various effects such as spray, range, etc
  + public class Bullet extends Sprite {

private Vector2 vTargetPos;

private Vector2 vVelocity;

private float fSpeed;

public Bullet(float x, float y, Texture tex, float fSpeed) {

super(tex);

vVelocity = Vector2.*Zero*;

this.fSpeed = fSpeed;

setPosition(x, y);

setOriginCenter();

setBounds(x, y, 8,8);

}

public void setTargetPos(Vector2 value, float spray) {

vTargetPos = value;

float fAngle = (float) Math.*toDegrees*(MathUtils.*atan2*(vTargetPos.y - getY(), vTargetPos.x - getX()));

int nSprayChooser = MathUtils.*random*(100);

float fSprayAmount = MathUtils.*random*(spray);

if (nSprayChooser != 0 && nSprayChooser < 50) {

fAngle += fSprayAmount;

} else if (nSprayChooser >= 50) {

fAngle -= fSprayAmount;

}

setRotation(fAngle);

vVelocity = new Vector2(fSpeed \* MathUtils.*cos*((float) Math.*toRadians*(fAngle)), fSpeed \* MathUtils.*sin*((float) Math.*toRadians*(fAngle)));

}

public void setTargetPos(float x, float y, float spray) {

setTargetPos(new Vector2(x, y), spray);

}

public void update() {

setPosition(getX() + vVelocity.x \* Gdx.*graphics*.getDeltaTime(), getY() + vVelocity.y \* Gdx.*graphics*.getDeltaTime());

}

public float findDistance(Vector2 vPos) {

return (float) Math.*sqrt*(Math.*pow*((getX() - vPos.x), 2) + Math.*pow*((getY() - vPos.y), 2));

}

* Use of JSON files for guns with different specs(attack speed, bullet speed, range, spray)
  + Json json = new Json();

gun = json.fromJson(Gun.class, Gdx.*files*.internal("json/revolver.json"));

fAttackCooldown = gun.getAttackSpeed();

* Significantly reduced amount of code needed for enemies by using vectors
  + public void move() {

vVelocity = Vector2.*Zero*;

fAngle = MathUtils.*random*(0f, 6.28319f);

vVelocity = new Vector2(fSpeed \* MathUtils.*cos*(fAngle), fSpeed \* MathUtils.*sin*(fAngle));

setPosition(getX() + vVelocity.x \* Gdx.*graphics*.getDeltaTime(), getY() + vVelocity.y \* Gdx.*graphics*.getDeltaTime());

}

Major Challenges/setbacks( reference specific code please):

* Player movement is currently does not feel good, very confusing and impractical. (W moves towards cursor instead of N)
  + public void move(int nDirection, float fSpeed) {

switch (nDirection) {

case 0:

vVelocity = (new Vector2(fSpeed \* MathUtils.*cos*(fAngle), 300 \* MathUtils.*sin*(fAngle)));

break;

case 1:

vVelocity = (new Vector2(-fSpeed \* MathUtils.*cos*(fAngle), -300 \* MathUtils.*sin*(fAngle)));

break;

case 2:

vVelocity = (new Vector2(fSpeed \* MathUtils.*cos*((float) (fAngle + 1.5708)), 300 \* MathUtils.*sin*((float) (fAngle + 1.5708))));

break;

case 3:

vVelocity = (new Vector2(fSpeed \* MathUtils.*cos*((float) (fAngle - 1.5708)), 300 \* MathUtils.*sin*((float) (fAngle - 1.5708))));

}

}

* Bullets to work
* Enemy movement was hard to change and inefficient so it was switched to vectors
  + It used to go through a bunch of if statements and generate a random number to decide if it was going to move N, S, E, W, then it would do the math for the angle. Now It moves randoming in any 360 degrees angle i just a few lines of code

Any modifications to your specifications/release schedule:

We went back and changed old code so we didn’t get as many releases done as we would have liked

We changed the cursor which was not planned, it only took a few lines of code:

Pixmap pm = new Pixmap(Gdx.*files*.internal("cursor.png"));

Gdx.*graphics*.setCursor(Gdx.*graphics*.newCursor(pm, 0, 0));

pm.dispose();

**Description of your scratch/test program:**

Describe the generic concept you needed to test out:

Moving the player towards the mouse

Source any web site/book that helped you with that concept:

StackOverflow: https://stackoverflow.com

Describe the code and the lesson that you learned from it:

The angle between the player’s position and the mouse are calculated using the Math.atan2() function

A velocity vector is created which multiplies a speed variable by the trig ratios of the angle to calculate its velocity along the X and Y axis and the player’s X and Y coordinates are updated using the velocity vector.

Describe any challenges that you enjoyed in integrating this scratch code into your major project: The player’s movement doesn’t quite feel right.

**Peer Assessment:**

Ethan: 100

John 100

Danny: 100