```
var isEnraged = false;

class Castle extends Entity{

    constructor(canvasX, canvasY){
        super(canvasX, canvasY, "resources/castle.png");

        this.healthCap = 20;
        this.internalHealth = this.healthCap;

        this.size = 200;

        //this.place(- canvasX / 2, - canvasY / 2);
        this.duration = 300;
    }

    static get isEnraged(){
        return isEnraged;
    }

    setStatus(enrage){
        isEnraged = enrage;
    }

    toggleStatus(){
        isEnraged = !isEnraged;
    }

    update(zombies){
        super.update();
        this.isHit(zombies);

        if(this.duration <= 0){
            this.setStatus(false);
        }

        if(isEnraged){
            this.image.src = "resources/enraged castle.png";
            this.duration -= 0.4;
        }
        else{
            this.image.src = "resources/castle.png";

            if(this.duration < 300)
                this.duration += 0.5;
        }

        this.drawRectangle(-700, 175, this.duration, 20,
Utility.greenToRedGradient(Math.floor(this.duration), 300));
    }

    isHit(zombies){
        for(zombie of zombies){

            if(zombie.isAlive && !zombie.isBinded && Utility.getDistance(this,
zombie) < 50){
                zombie.kill();
                this.internalHealth--;
            }
```

```
        }
    }
}

//============================================================================
class Entity{

    constructor(canvasX, canvasY, pathToImage){
        this.context = null;
        this.canvas = null;
        this.id = 0;

        this.x = 0;
        this.y = 0;

        this.canvasX = canvasX;
        this.canvasY = canvasY;

        this.image = new Image();
        this.image.src = pathToImage;

        this.isAlive = true;

        this.isBinded = false;

        this.size = 50;

        this.healthCap = 5;
        this.internalHealth = this.healthCap;

        this.speed = 0.15;
    }

    initialize(context, canvas){
        this.context = context;
        this.canvas = canvas;
    }

    respawn(place){
        if(place){
            this.x = Math.floor(Math.random() * this.canvasX) - (this.canvasX / 2);
            this.y = Math.floor(Math.random() * this.canvasY) - (this.canvasY / 2);
        }

        this.isAlive = true;
        this.internalHealth = this.healthCap;
    }

    respawnAround(angle, radius, focal){
        var displacementX = Math.cos(angle) * radius;
        var displacementY = Math.sin(angle) * radius;

        this.respawn(false);
        this.place(focal.x + displacementX, focal.y + displacementY);
    }

    update(){
        if(this.internalHealth <= 0)
            this.isAlive = false;
```

```
        else
            this.isAlive = true;

        this.place(this.x, this.y);
    }

    kill(){
        this.internalHealth = 0;
        this.isAlive = false;
    }

    scatter(){
        this.place(
            (Math.random() * this.canvasX) - (this.canvasX / 2),
            (Math.random() * this.canvasY) - (this.canvasY / 2)
        );
    }

    place(newX, newY){
        if(this.isAlive){
            this.x = newX;
            this.y = newY;

            this.context.drawImage(
                this.image,
                (this.x + (this.canvasX / 2)) - (this.size / 2),
                -(this.y - (this.canvasY / 2)) - (this.size / 2),
                this.size,
                this.size
                );

            //this.writeTextCartesian(this.id, this.x -5, this.y + 50);
            this.drawRectangle(this.x - (this.size / 2), this.y - 40,
this.internalHealth * 10, 5, "#FF0000");
        }
    }

    pursuit(object){
        var deltaX = this.x - object.x;
        var deltaY = this.y - object.y;

        var angle = Math.PI + Math.atan2(deltaY, deltaX);

        var xSpeed = Math.cos(angle) * this.speed;
        var ySpeed = Math.sin(angle) * this.speed;


        this.place(
            xSpeed + this.x,
            ySpeed + this.y
        );
    }


    getDistanceTo(object){
        return Utility.getDistance(object, this);
    }

    writeText(text, x, y){
```

```
        context.font = "30px Arial";
        context.fillStyle = "#FF0000";
        context.fillText(text, x, y);
    }

    writeTextCartesian(text, x, y){
        writeText(
            text,
            x + (canvasBoundX / 2),
            -(y - (canvasBoundY / 2))
        )
    }

    drawRectangle(x, y, width, height, color){
        this.context.fillStyle = color;
        this.context.fillRect(
            Utility.toCartesianX(x, this.canvasX),
            Utility.toCartesianY(y, this.canvasY),
            width,
            height);
    }


}

//=================================================================
class Graveyard extends Entity{

    constructor(canvasX, canvasY){
        super(canvasX, canvasY, "resources/grave.png");

        this.healthCap = 20;
        this.internalHealth = this.healthCap;

        this.size = 150;

        //this.place(- canvasX / 2, - canvasY / 2);
    }

    update(knights){
        super.update();
        this.isHit(knights);
    }

    isHit(knights){
        for(knight of knights){

            if(knight.isAlive && Utility.getDistance(this, knight) < 50){
                knight.kill();
                this.internalHealth--;
            }
        }
    }
}
//=============================================================

class Knight extends Entity{

    constructor(canvasBoundX, canvasBoundY){
```

```
    super(canvasBoundX, canvasBoundY, "resources/knight.png");

    this.healthCap = 3;
    this.internalHealth = 3;
    this.speed = 0.2;
}

update(zombies, graveyard){
    super.update();

    if(!this.isBinded){
        this.punch(zombies, graveyard);

        if(Castle.isEnraged){
            this.speed = 1;
            this.image.src = "resources/enraged knight.png";
            this.size = 60;
        }
        else{
            this.size = 50;
            this.speed = 0.2;
            this.image.src = "resources/knight.png";
        }


    }
    else{
        this.size = 75;
    }
}

punch(zombies, graveyard){
    var closest = {distance: 10000, id: -1};

    for(zombie of zombies){
        if(!zombie.isAlive || zombie.isBinded)
            continue;

        var distance = Utility.getDistance(this, zombie);

        if(distance < closest.distance){
            closest.distance = distance;
            closest.id = zombie.id;
        }
    }

    if(closest.id != -1){
        if(Utility.getDistance(this, graveyard) > 300)
            super.pursuit(zombies[closest.id]);
        else
            super.pursuit(graveyard);
    }

    else{
        super.pursuit(graveyard);
    }

    if(closest.distance < 5){
        zombies[closest.id].kill();
```

```
            if(!Castle.isEnraged)
                this.internalHealth--;
            else
                this.internalHealth -= 0.2;
        }
    }
}
//=============================================================
class Meat{

    constructor(canvasBoundX, canvasBoundY){
        this.canvas = null;
        this.context = null;

        this.canvasBoundX = canvasBoundX;
        this.canvasBoundY = canvasBoundY;

        this.x = Math.floor(Math.random() * this.canvasBoundX) - (this.canvasBoundX
/ 2);
        this.y = Math.floor(Math.random() * this.canvasBoundY) - (this.canvasBoundY
/ 2);

        this.image = new Image();

        this.image.src = "resources/meat.png";
    }

    update(){
        this.place(this.x, this.y);
    }

    respawn(){
        this.place(
            Math.floor(Math.random() * this.canvasBoundX) - (this.canvasBoundX /
2),
            Math.floor(Math.random() * this.canvasBoundY) - (this.canvasBoundY / 2)
        );

        this.place(this.x, this.y);
    }

    place(x, y){
        this.x = x;
        this.y = y;

        this.context.drawImage(
            this.image,
            (this.x + (this.canvasBoundX / 2)) - 25,
            -(this.y - (this.canvasBoundY / 2)) - 25,
            50,
            50
            );
    }



}
```

```
//=================================================================
class Mouse{

    constructor(x, y){
        this.x = x;
        this.y = y;

        this.clicked = false;
        this.bindID = -1;
        this.isBinded = false;
        this.mode = "zombie";
    }

    isClickedOn(object){
        return this.clicked && this.isHoveringOver(object);
    }

    isHoveringOver(object){
        return Utility.getDistance(this, object) < 50;
    }

    toggleMode(){
        if(this.mode == "zombie")
            this.mode = "knight";
        else if(this.mode == "knight")
            this.mode = "turret";
        else if(this.mode == "turret")
            this.mode = "zombie";
    }
}
//=================================================================

class Turret extends Entity{

    constructor(canvasX, canvasY){
        super(canvasX, canvasY, "resources/turret.png");

        this.size = 75;

        this.healthCap = 10;
        this.internalHealth = 10;

        this.arrow = new Arrow(0, 0 , 1, 5);
    }

    update(zombies){
        super.update();

        if(!this.isBinded){
            this.size = 75;
        }
        else{
            this.size = 100;
        }

        if(this.isAlive){
            this.track(zombies);
        }
```

```
        this.arrow.follow(this, zombies);
    }

    respawn(place){
        super.respawn(place);
    }

    place(newX, newY){
        super.place(newX, newY);

        if(!this.arrow.isActive)
            this.arrow.reset(this.x, this.y);

    }

    initialize(context, canvas){
        this.context = context;
        this.canvas = canvas;

        this.arrow.context = context;

        this.arrow.reset(this.x, this.y);
    }

    track(zombies){
        var closest = {distance: 10000, id: -1};

        for(zombie of zombies){
            if(!zombie.isAlive)
                continue;

            var distance = Utility.getDistance(this, zombie);

            if(distance < closest.distance){
                closest.distance = distance;
                closest.id = zombie.id;
            }
        }

        if(closest.id != -1){

            if(!this.arrow.isActive){
                this.internalHealth -= 0.5;
                this.arrow.isActive = true;
                this.arrow.setTarget(zombies[closest.id]);
            }

        }
    }

}

class Arrow{
    constructor(x, y, speed, size){
        this.x = x;
        this.y = y;
        this.speed = speed;
        this.size = size;
```

```
        this.context = null;

        this.isActive = false;

        this.target = {angle: 0, id: -1};
    }

    follow(turret, zombies){

        if(this.target.id != -1){
            var xSpeed = Math.cos(this.target.angle) * this.speed;
            var ySpeed = Math.sin(this.target.angle) * this.speed;

            if(turret.isAlive){
                if(this.isActive){
                    this.place(
                        xSpeed + this.x,
                        ySpeed + this.y,
                        turret.canvasX,
                        turret.canvasY
                    );
                }

                for(zombie of zombies){

                    if(!zombie.isAlive)
                        continue;

                    if(Utility.getDistance(this, zombie) < 25){
                        zombie.internalHealth--;
                        this.reset(turret.x, turret.y);
                        break;
                    }
                }

                if(Math.abs(this.x) > turret.canvasX / 2 || Math.abs(this.y) >
turret.canvasY / 2){
                    this.reset(turret.x, turret.y);
                }
            }
        }
    }

    setTarget(zombie){
        this.target.id = zombie.id;

        var deltaX = this.x - zombie.x;
        var deltaY = this.y - zombie.y;

        var angle = Math.PI + Math.atan2(deltaY, deltaX);

        this.target.angle = angle;
    }

    place(x, y, maxX, maxY){
        this.x = x;
        this.y = y;

        this.context.fillStyle = "#FF0000";
```

```
        this.context.beginPath();
        this.context.arc(
            Utility.toCartesianX(this.x, maxX),
            Utility.toCartesianY(this.y, maxY),
            this.size, 0, 2 * Math.PI, true);
            this.context.closePath();
            this.context.fill();
    }

    reset(x, y){
        this.x = x;
        this.y = y;

        this.isActive = false;
    }
}
//======================================================================
class Zombie extends Entity{

    constructor(canvasBoundX, canvasBoundY){
        super(canvasBoundX, canvasBoundY, "resources/skull.png");

        this.kill();
    }

    update(object){
        if(this.internalHealth <= 0)
            this.isAlive = false;
        else
            this.isAlive = true;

        if(this.isAlive){
            if(!this.isBinded){
                this.pursuit(object);
                this.size = 50;
            }
            else{
                this.size = 75;
            }
        }
    }


}
//=======================================
var canvas = null;
var context = null;

var canvasBoundX = 1450;
var canvasBoundY = 600;

var xOffset = 50;
var yOffset = 50;

let timer = setInterval("update()", 1);
var isOn = false;
var ticks = 0;
var level = 1;
```

```
var respawnBehavior = "random";

const mouse = new Mouse(0, 0);
const castle = new Castle(canvasBoundX, canvasBoundY);
const graveyard = new Graveyard(canvasBoundX, canvasBoundY);

//===zombies=====
var zombieCap = 100;
const zombies = new Array();

class Manager{
    static get TIMESTAMP(){
        return ticks;
    }

    static get LEVEL(){
        return level;
    }
}

for(var i = 0; i < zombieCap; i++){
    zombies[i] = new Zombie(canvasBoundX, canvasBoundY);
    zombies[i].id = i;
}

//=====knights====
var knightCap = 20;
const knights = new Array();

for(var i = 0; i < knightCap; i++){
    knights[i] = new Knight(canvasBoundX, canvasBoundY);
    knights[i].id = i;
}

//======turrets=====
var turretCap = 5;
const turrets = new Array();

for(var i = 0; i < turretCap; i++){
    turrets[i] = new Turret(canvasBoundX, canvasBoundY);
    turrets[i].id = i;
}

//==========Input Handles===============
window.addEventListener("keypress", function(event){

    switch(event.key){
        case "q":
            slap();
            break;

        case " ":
            respawnBehavior = "random";
            respawnManually();
            break;

        case "r":
            mouse.toggleMode();
            break;
```

```
        case "e":
            respawnBehavior = "mouse";
            respawnManually();
            break;

        case "t":
            castle.toggleStatus();
            break;

        case "f":
            scatterKnights();
            break;

    }
},
true);

function respawn(){
    if(ticks % 50 == 0){
        for(zombie of zombies){
            if(!zombie.isAlive){
                zombie.respawnAround(
                    Math.random() * Math.PI + (Math.PI * 3/4),
                    300,
                    graveyard);
                break;
            }
        }
    }

    var knightProc = Castle.isEnraged ? 100 : 200; //spawn faster if its enraged

    if(ticks % knightProc == 0){
        for(knight of knights){
            if(!knight.isAlive){
                knight.respawnAround(
                    Math.random() * Math.PI - (Math.PI * 1/4),
                    200,
                    castle);
                break;
            }
        }
    }
}

function respawnManually(){
    if(mouse.mode == "zombie"){
        for(zombie of zombies){
            if(!zombie.isAlive){
                if(respawnBehavior == "random")
                    zombie.respawn(true);

                else if(respawnBehavior == "mouse"){
                    zombie.respawn(false)
                    zombie.place(mouse.x, mouse.y);
                }
                break;
            }
```

```
        }
    }
    else if(mouse.mode == "knight"){
        for(knight of knights){
            if(!knight.isAlive){
                if(respawnBehavior == "random")
                    knight.respawn(true);
                else if(respawnBehavior == "mouse"){
                    knight.respawn(false);
                    knight.place(mouse.x, mouse.y);
                }
                break;
            }
        }
    }
    else if(mouse.mode == "turret"){
        for(turret of turrets){
            if(!turret.isAlive){
                if(respawnBehavior == "random")
                    turret.respawn(true);
                else if(respawnBehavior == "mouse"){
                    turret.respawn(false);
                    turret.place(mouse.x, mouse.y);
                }
                break;
            }
        }
    }
}

function scatterKnights(objects){
    for(knight of knights){
        if(!knight.isAlive)
            continue;

        knight.scatter();
    }
}

function slap(){
    for(zombie of zombies){
        if(mouse.isHoveringOver(zombie) && zombie.isAlive){
            zombie.internalHealth -= 1;
            break;
        }
    }
}

function toggleBehavior(){
    if(respawnBehavior == "random")
        respawnBehavior = "mouse";
    else
        respawnBehavior = "random";
}

function whileClick(){
    mouse.clicked = true;
}
```

```
function whileNotClicked(){
    mouse.clicked = false;
}

function updateMouse(event){
    mouse.x = (event.clientX - xOffset) - (canvasBoundX / 2);
    mouse.y = -((event.clientY - yOffset) - (canvasBoundY / 2));
}
//=======================================

function update(){
    ticks += 1;

    resetBackground();
    updateEntities();
    checkState();
    logData();
    respawn();
    checkMouse();
}

function initialize(){
    canvas = document.getElementById("myCanvas");
    context = canvas.getContext("2d");

    for(zombie of zombies){
        zombie.initialize(context, canvas);
        zombie.kill();
    }

    for(knight of knights){
        knight.initialize(context, canvas);
        knight.kill();
    }

    for(turret of turrets){
        turret.initialize(context, canvas);
        turret.kill();
    }

    castle.initialize(context, canvas);
    castle.place(-canvasBoundX / 2 + 100, -canvasBoundY / 2 + 100);

    graveyard.initialize(context, canvas);
    graveyard.place(canvasBoundX / 2 - 75, canvasBoundY / 2 - 75);

    //turnOn();
    resetBackground();
}

function checkMouse(){

    if(mouse.mode == "zombie"){
        for(zombie of zombies){
            if(mouse.isClickedOn(zombie) && zombie.isAlive && !mouse.isBinded){
                mouse.isBinded = true;
                mouse.bindID = zombie.id;
                zombie.isBinded = true;
                break;
```

```
            }
        }

        if(mouse.clicked && mouse.isBinded && mouse.bindID >= 0)
            zombies[mouse.bindID].place(mouse.x, mouse.y);
        else{
            if(mouse.bindID >= 0)
                zombies[mouse.bindID].isBinded = false;
            mouse.bindID = -1;
            mouse.isBinded = false;
        }
    }

    else if(mouse.mode == "knight"){
        for(knight of knights){
            if(mouse.isClickedOn(knight) && knight.isAlive && !mouse.isBinded){
                mouse.isBinded = true;
                mouse.bindID = knight.id;
                knight.isBinded = true;
                break;
            }
        }

        if(mouse.clicked && mouse.isBinded && mouse.bindID >= 0)
            knights[mouse.bindID].place(mouse.x, mouse.y);
        else{
            if(mouse.bindID >= 0)
                knights[mouse.bindID].isBinded = false;
            mouse.bindID = -1;
            mouse.isBinded = false;
        }
    }

    else if(mouse.mode == "turret"){
        for(turret of turrets){
            if(mouse.isClickedOn(turret) && turret.isAlive && !mouse.isBinded){
                mouse.isBinded = true;
                mouse.bindID = turret.id;
                turret.isBinded = true;
                break;
            }
        }

        if(mouse.clicked && mouse.isBinded && mouse.bindID >= 0)
            turrets[mouse.bindID].place(mouse.x, mouse.y);
        else{
            if(mouse.bindID >= 0)
                turrets[mouse.bindID].isBinded = false;
            mouse.bindID = -1;
            mouse.isBinded = false;
        }
    }
}

function checkState(){
    if(!castle.isAlive){
        turnOff();
        resetBackground();
        writeTextCartesian("Zombies Win!", 0, 0);
```

```
        writeTextCartesian("Final Score: " + ticks, 0, 100);
    }

    else if(!graveyard.isAlive){
        //turnOff();
        //resetBackground();
        //writeTextCartesian("Knights Win!", 0, 0);
        for(zombie of zombies){
            zombie.healthCap += 5;
            zombie.speed += 0.25;
        }

        reset();
        level++;
    }
}

function reset(){
    for(zombie of zombies){
        zombie.kill();
    }

    for(knight of knights){
        knight.kill();
    }

    for(turret of turrets){
        turret.kill();
    }

    //castle.internalHealth = castle.healthCap;
    graveyard.internalHealth = graveyard.healthCap;

    turnOn();
}

function logData(){
    /*
    var knightsAlive = 0;

    for(knight of knights){
        if(knight.isAlive)
            knightsAlive++;
    }

    writeText("Knights Alive: " + knightsAlive, 50, 50);

    //=======================================
    var zombiesAlive = 0;

    for(zombie of zombies){
        if(zombie.isAlive)
            zombiesAlive++;
    }

    writeText("Zombies Alive: " + zombiesAlive, 50, 100);
    //=======================================
    var turretsAlive = 0;
```

```
    for(turret of turrets){
        if(turret.isAlive)
            turretsAlive++;
    }

    writeText("Turrets Alive: " + turretsAlive, 50, 150);
    */

    //=========================================
    writeText("Mode: " + mouse.mode, 10, 50);
    writeText("Behavior: " + respawnBehavior, 10, 100);
    writeText("Ticks: " + ticks, 10, 200);
    writeText("Level: " + Manager.LEVEL, 1300, 550);
}

function updateEntities(){
    castle.update(zombies);
    graveyard.update(knights);

    for(zombie of zombies){
        zombie.update(castle);
    }

    for(knight of knights){
        knight.update(zombies, graveyard);
    }

    for(turret of turrets){
        turret.update(zombies);
    }
}

function writeText(text, x, y){
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");

    context.font = "30px Arial";
    context.fillStyle = "#FF0000";
    context.fillText(text, x, y);
}

function writeTextCartesian(text, x, y){
    //(this.x + (this.canvasX / 2)) - (this.size / 2),
    //-(this.y - (this.canvasY / 2)) - (this.size / 2),
    writeText(
        text,
        x + (canvasBoundX / 2),
        -(y - (canvasBoundY / 2))
    )
}

function resetBackground(){
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    //paint the background of the canvas
    context.fillStyle="#FBEEAC"; //"#ADD8E6";
    context.fillRect(0, 0, canvasBoundX, canvasBoundY);
}
```

```
function toggleTimer(){
    //if the timer is on the turn it off, if its not then turn it on
    isOn ? turnOff() : turnOn();
}

//manually turns on the timer
function turnOn(){
    turnOff();
    //turns the timer on
    timer = setInterval("update()", 1);
    isOn = true;
}

//manually turns off the timer
function turnOff(){
    //turns the timer off
    clearInterval(timer);
    isOn = false;
}
```