Laporan Resmi Source Code Game Analisis



Oleh :

Muhammad Shidqi Firdaus 4210161006

**D4 TEKNOLOGI GAME**

**POLITEKNIK ELEKTRONIKA NEGERI SURABAYA**

**SURABAYA**

**2017**

Game yang dianalisis adalah alien invasion yang telah dibuat ulang menggunakan HTML5

sumber: <https://github.com/cykod/AlienInvasion>

Game class

|  |  |
| --- | --- |
| Var sprites { |  |
|  | ship: { sx: 0, sy: 0, w: 37, h: 42, frames: 1 }, |
|  | missile: { sx: 0, sy: 30, w: 2, h: 10, frames: 1 }, |
|  | enemy\_purple: { sx: 37, sy: 0, w: 42, h: 43, frames: 1 }, |
|  | enemy\_bee: { sx: 79, sy: 0, w: 37, h: 43, frames: 1 }, |
|  | enemy\_ship: { sx: 116, sy: 0, w: 42, h: 43, frames: 1 }, |
|  | enemy\_circle: { sx: 158, sy: 0, w: 32, h: 33, frames: 1 }, |
|  | explosion: { sx: 0, sy: 64, w: 64, h: 64, frames: 12 }, |
|  | enemy\_missile: { sx: 9, sy: 42, w: 3, h: 20, frame: 1, } |
|  | }; |
|  |  |
|  | var enemies = { |
|  | straight: { x: 0, y: -50, sprite: 'enemy\_ship', health: 10, |
|  | E: 100 }, |
|  | ltr: { x: 0, y: -100, sprite: 'enemy\_purple', health: 10, |
|  | B: 75, C: 1, E: 100, missiles: 2 }, |
|  | circle: { x: 250, y: -50, sprite: 'enemy\_circle', health: 10, |
|  | A: 0, B: -100, C: 1, E: 20, F: 100, G: 1, H: Math.PI/2 }, |
|  | wiggle: { x: 100, y: -50, sprite: 'enemy\_bee', health: 20, |
|  | B: 50, C: 4, E: 100, firePercentage: 0.001, missiles: 2 }, |
|  | step: { x: 0, y: -50, sprite: 'enemy\_circle', health: 10, |
|  | B: 150, C: 1.2, E: 75 } |
|  | }; |
|  | Function diatas digunakan untuk memberikan atrribut kepada objek |
|  | var OBJECT\_PLAYER = 1, |
|  | OBJECT\_PLAYER\_PROJECTILE = 2, |
|  | OBJECT\_ENEMY = 4, |
|  | OBJECT\_ENEMY\_PROJECTILE = 8, |
|  | OBJECT\_POWERUP = 16; |
|  |  |
|  | var startGame = function() { |
|  | var ua = navigator.userAgent.toLowerCase(); |
|  |  |
|  | // Only 1 row of stars |
|  | if(ua.match(/android/)) { |
|  | Game.setBoard(0,new Starfield(50,0.6,100,true)); |
|  | } else { |
|  | Game.setBoard(0,new Starfield(20,0.4,100,true)); |
|  | Game.setBoard(1,new Starfield(50,0.6,100)); |
|  | Game.setBoard(2,new Starfield(100,1.0,50)); |
|  | } |
|  | Game.setBoard(3,new TitleScreen("Alien Invasion", |
|  | "Press fire to start playing", |
|  | playGame)); |
|  | }; |
|  |  |
|  | var level1 = [ |
|  | // Start, End, Gap, Type, Override |
|  | [ 0, 4000, 500, 'step' ], |
|  | [ 6000, 13000, 800, 'ltr' ], |
|  | [ 10000, 16000, 400, 'circle' ], |
|  | [ 17800, 20000, 500, 'straight', { x: 50 } ], |
|  | [ 18200, 20000, 500, 'straight', { x: 90 } ], |
|  | [ 18200, 20000, 500, 'straight', { x: 10 } ], |
|  | [ 22000, 25000, 400, 'wiggle', { x: 150 }], |
|  | [ 22000, 25000, 400, 'wiggle', { x: 100 }] |
|  | ]; |
|  |  |
|  |  |
|  |  |
|  | var playGame = function() { |
|  | var board = new GameBoard(); |
|  | board.add(new PlayerShip()); |
|  | board.add(new Level(level1,winGame)); |
|  | Game.setBoard(3,board); |
|  | Game.setBoard(5,new GamePoints(0)); |
|  | }; |
|  |  |
|  | var winGame = function() { |
|  | Game.setBoard(3,new TitleScreen("You win!", |
|  | "Press fire to play again", |
|  | playGame)); |
|  | }; |
|  |  |
|  | var loseGame = function() { |
|  | Game.setBoard(3,new TitleScreen("You lose!", |
|  | "Press fire to play again", |
|  | playGame)); |
|  | }; |
|  |  |
|  | var Starfield = function(speed,opacity,numStars,clear) { |
|  |  |
|  | // Set up the offscreen canvas |
|  | var stars = document.createElement("canvas"); |
|  | stars.width = Game.width; |
|  | stars.height = Game.height; |
|  | var starCtx = stars.getContext("2d"); |
|  |  |
|  | var offset = 0; |
|  |  |
|  | // If the clear option is set, |
|  | // make the background black instead of transparent |
|  | if(clear) { |
|  | starCtx.fillStyle = "#000"; |
|  | starCtx.fillRect(0,0,stars.width,stars.height); |
|  | } |
|  |  |
|  | // Now draw a bunch of random 2 pixel |
|  | // rectangles onto the offscreen canvas |
|  | starCtx.fillStyle = "#FFF"; |
|  | starCtx.globalAlpha = opacity; |
|  | for(var i=0;i<numStars;i++) { |
|  | starCtx.fillRect(Math.floor(Math.random()\*stars.width), |
|  | Math.floor(Math.random()\*stars.height), |
|  | 2, |
|  | 2); |
|  | } |
|  |  |
|  | // This method is called every frame |
|  | // to draw the starfield onto the canvas |
|  | this.draw = function(ctx) { |
|  | var intOffset = Math.floor(offset); |
|  | var remaining = stars.height - intOffset; |
|  |  |
|  | // Draw the top half of the starfield |
|  | if(intOffset > 0) { |
|  | ctx.drawImage(stars, |
|  | 0, remaining, |
|  | stars.width, intOffset, |
|  | 0, 0, |
|  | stars.width, intOffset); |
|  | } |
|  |  |
|  | // Draw the bottom half of the starfield |
|  | if(remaining > 0) { |
|  | ctx.drawImage(stars, |
|  | 0, 0, |
|  | stars.width, remaining, |
|  | 0, intOffset, |
|  | stars.width, remaining); |
|  | } |
|  | }; |
|  |  |
|  | // This method is called to update |
|  | // the starfield |
|  | this.step = function(dt) { |
|  | offset += dt \* speed; |
|  | offset = offset % stars.height; |
|  | }; |
|  | }; |
|  |  |
|  | var PlayerShip = function() { |
|  | this.setup('ship', { vx: 0, reloadTime: 0.25, maxVel: 200 }); |
|  |  |
|  | this.reload = this.reloadTime; |
|  | this.x = Game.width/2 - this.w / 2; |
|  | this.y = Game.height - Game.playerOffset - this.h; |
|  |  |
|  | this.step = function(dt) { |
|  | if(Game.keys['left']) { this.vx = -this.maxVel; } |
|  | else if(Game.keys['right']) { this.vx = this.maxVel; } |
|  | else { this.vx = 0; } |
|  |  |
|  | this.x += this.vx \* dt; |
|  |  |
|  | if(this.x < 0) { this.x = 0; } |
|  | else if(this.x > Game.width - this.w) { |
|  | this.x = Game.width - this.w; |
|  | } |
|  |  |
|  | this.reload-=dt; |
|  | if(Game.keys['fire'] && this.reload < 0) { |
|  | Game.keys['fire'] = false; |
|  | this.reload = this.reloadTime; |
|  |  |
|  | this.board.add(new PlayerMissile(this.x,this.y+this.h/2)); |
|  | this.board.add(new PlayerMissile(this.x+this.w,this.y+this.h/2)); |
|  | } |
|  | }; |
|  | }; |
|  |  |
|  | PlayerShip.prototype = new Sprite(); |
|  | PlayerShip.prototype.type = OBJECT\_PLAYER; |
|  |  |
|  | PlayerShip.prototype.hit = function(damage) { |
|  | if(this.board.remove(this)) { |
|  | loseGame(); |
|  | } |
|  | }; |
|  |  |
|  |  |
|  | var PlayerMissile = function(x,y) { |
|  | this.setup('missile',{ vy: -700, damage: 10 }); |
|  | this.x = x - this.w/2; |
|  | this.y = y - this.h; |
|  | }; |
|  |  |
|  | PlayerMissile.prototype = new Sprite(); |
|  | PlayerMissile.prototype.type = OBJECT\_PLAYER\_PROJECTILE; |
|  |  |
|  | PlayerMissile.prototype.step = function(dt) { |
|  | this.y += this.vy \* dt; |
|  | var collision = this.board.collide(this,OBJECT\_ENEMY); |
|  | if(collision) { |
|  | collision.hit(this.damage); |
|  | this.board.remove(this); |
|  | } else if(this.y < -this.h) { |
|  | this.board.remove(this); |
|  | } |
|  | }; |
|  |  |
|  |  |
|  | var Enemy = function(blueprint,override) { |
|  | this.merge(this.baseParameters); |
|  | this.setup(blueprint.sprite,blueprint); |
|  | this.merge(override); |
|  | }; |
|  |  |
|  | Enemy.prototype = new Sprite(); |
|  | Enemy.prototype.type = OBJECT\_ENEMY; |
|  |  |
|  | Enemy.prototype.baseParameters = { A: 0, B: 0, C: 0, D: 0, |
|  | E: 0, F: 0, G: 0, H: 0, |
|  | t: 0, reloadTime: 0.75, |
|  | reload: 0 }; |
|  |  |
|  | Enemy.prototype.step = function(dt) { |
|  | this.t += dt; |
|  |  |
|  | this.vx = this.A + this.B \* Math.sin(this.C \* this.t + this.D); |
|  | this.vy = this.E + this.F \* Math.sin(this.G \* this.t + this.H); |
|  |  |
|  | this.x += this.vx \* dt; |
|  | this.y += this.vy \* dt; |
|  |  |
|  | var collision = this.board.collide(this,OBJECT\_PLAYER); |
|  | if(collision) { |
|  | collision.hit(this.damage); |
|  | this.board.remove(this); |
|  | } |
|  |  |
|  | if(Math.random() < 0.01 && this.reload <= 0) { |
|  | this.reload = this.reloadTime; |
|  | if(this.missiles == 2) { |
|  | this.board.add(new EnemyMissile(this.x+this.w-2,this.y+this.h)); |
|  | this.board.add(new EnemyMissile(this.x+2,this.y+this.h)); |
|  | } else { |
|  | this.board.add(new EnemyMissile(this.x+this.w/2,this.y+this.h)); |
|  | } |
|  |  |
|  | } |
|  | this.reload-=dt; |
|  |  |
|  | if(this.y > Game.height || |
|  | this.x < -this.w || |
|  | this.x > Game.width) { |
|  | this.board.remove(this); |
|  | } |
|  | }; |
|  |  |
|  | Enemy.prototype.hit = function(damage) { |
|  | this.health -= damage; |
|  | if(this.health <=0) { |
|  | if(this.board.remove(this)) { |
|  | Game.points += this.points || 100; |
|  | this.board.add(new Explosion(this.x + this.w/2, |
|  | this.y + this.h/2)); |
|  | } |
|  | } |
|  | }; |
|  |  |
|  | var EnemyMissile = function(x,y) { |
|  | this.setup('enemy\_missile',{ vy: 200, damage: 10 }); |
|  | this.x = x - this.w/2; |
|  | this.y = y; |
|  | }; |
|  |  |
|  | EnemyMissile.prototype = new Sprite(); |
|  | EnemyMissile.prototype.type = OBJECT\_ENEMY\_PROJECTILE; |
|  |  |
|  | EnemyMissile.prototype.step = function(dt) { |
|  | this.y += this.vy \* dt; |
|  | var collision = this.board.collide(this,OBJECT\_PLAYER) |
|  | if(collision) { |
|  | collision.hit(this.damage); |
|  | this.board.remove(this); |
|  | } else if(this.y > Game.height) { |
|  | this.board.remove(this); |
|  | } |
|  | }; |
|  |  |
|  |  |
|  |  |
|  | var Explosion = function(centerX,centerY) { |
|  | this.setup('explosion', { frame: 0 }); |
|  | this.x = centerX - this.w/2; |
|  | this.y = centerY - this.h/2; |
|  | }; |
|  |  |
|  | Explosion.prototype = new Sprite(); |
|  |  |
|  | Explosion.prototype.step = function(dt) { |
|  | this.frame++; |
|  | if(this.frame >= 12) { |
|  | this.board.remove(this); |
|  | } |
|  | }; |
|  |  |
|  | window.addEventListener("load", function() { |
|  | Game.initialize("game",sprites,startGame); |
|  | }); |