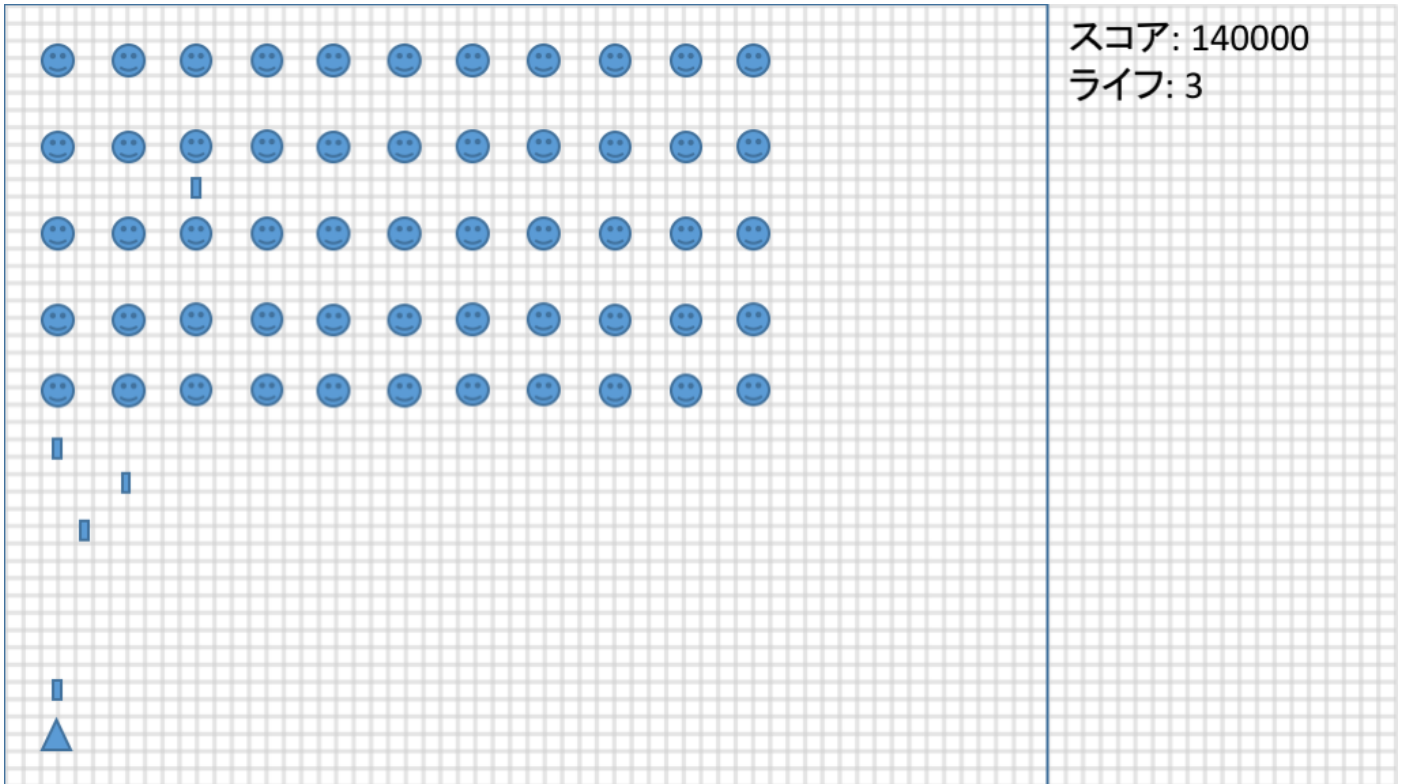


「レトロなシューティング」を作ろう

https://github.com/akokubo/retro_shooting

画面デザイン



1. ディスプレイ・ウィンドウのサイズを指定

https://github.com/akokubo/retro_shooting/tree/4cfe585d930c22bf3f907918e4420376a4328d45/retro_shooting

```
void setup() {  
  // ディスプレイ・ウィンドウのサイズを 640x360 に  
  size(640, 360);  
}  
  
void draw() {  
}
```

2. スプライト・クラスの作成

https://github.com/akokubo/retro_shooting/tree/f5f01fa240534d18d41d7db6f7264abf37576a3b/retro_shooting

Sprite タブ

```
// スプライト・クラス
class Sprite {
    // 画像
    PImage image;

    // 座標
    float x;
    float y;

    // コンストラクタ(デフォルト)
    Sprite() {
    }

    // コンストラクタ(画像を指定するとき)
    Sprite(PImage image) {
        this.image = image;
    }

    // コンストラクタ(画像と座標を指定するとき)
    Sprite(PImage image, float x, float y) {
        this.image = image;
        this.x = x;
        this.y = y;
    }

    // 表示
    void display() {
        image(this.image, x, y);
    }

    // 移動
    void move() {
    }

    // 当たり判定
    boolean isContactedWith(Sprite sprite) {
        // 判定結果を入れる変数。デフォルトは false
        boolean result = false;

        // 当たったら true に
        if (dist(x, y, sprite.x, sprite.y) < (image.width + sprite.image.width) / 2) {
            result = true;
        }

        return result;
    }
}
```

メイン・タブ

変更なし

3. 画像(エイリアン、爆弾、キャノン、レーザー、ステージ)の追加

https://github.com/akokubo/retro_shooting/tree/034b9b2319d5cec9c88f02a383f0b70e92d9de7f/retro_shooting

stage.png: 480x360 ピクセル



cannon.png: 16x16 ピクセル



laser.png: 4x16 ピクセル



alien.png: 16x16 ピクセル



bomb.png: 16x16 ピクセル



4. ステージ・クラスの作成

https://github.com/akokubo/retro_shooting/tree/cb6b090a96540a9d31f496e99f68b2f00432d304/retro_shooting

Stage タブ

```
// ステージ・クラス
class Stage extends Sprite {
    PFont font; // フォント

    // コンストラクタ
    Stage(PImage image) {
        super(image);

        // 画像の中心を座標として設定
        x = image.width / 2;
        y = image.height / 2;

        // フォントの生成
        font = createFont("MS Gothic", 20);
    }

    // 表示(オーバーライド)
    void display() {
        // 親クラスのメソッドをそのまま呼ぶ
        super.display();

        // フォントと色の設定
        textFont(font);
        fill(0);

        // スコアの表示(仮)
        text("SCORE: 14000", this.image.width + 20, 20);

        // ライフの表示(仮)
        text("LIFE: 3", this.image.width + 20, 40);
    }
}
```

メイン・タブ

```
// オブジェクト
Stage stage;

void setup() {
    // ディスプレイ・ウィンドウのサイズを 640x360 に
    size(640, 360);

    // 画像を中心に表示するモード
    imageMode(CENTER);

    // オブジェクトの生成
    stage = new Stage(loadImage("stage.png"));
}

void draw() {
    // 残像を消す
```

```
background(204);  
  
// 表示  
stage.display();  
}
```

Sprite タブ

変更なし

5. キャノン・クラスの実装

https://github.com/akokubo/retro_shooting/tree/daaecab9599453db35d3b3ef299f26c003b09f55/retro_shooting

Cannon タブ

```
// キャノン・クラス
class Cannon extends Sprite {
  // コンストラクタ
  Cannon(PImage image, float x, float y) {
    super(image, x, y);
  }

  // 移動(オーバーライド)
  void move() {
    x = mouseX;
  }
}
```

メイン・タブ

```
// オブジェクト
Stage stage;
Cannon cannon;

void setup() {
  // ディスプレイ・ウィンドウのサイズを 640x360 に
  size(640, 360);

  // 画像を中心に表示するモード
  imageMode(CENTER);

  // オブジェクトの生成
  stage = new Stage(loadImage("stage.png"));
  cannon = new Cannon(loadImage("cannon.png"),
    stage.image.width / 2, stage.image.height - 38);
}

void draw() {
  // 残像を消す
  background(204);

  // キャノンの移動
  cannon.move();

  // 表示
  stage.display();
  cannon.display();
}
```

Sprite、Stage タブ

変更なし

6. キャノンの移動速度の上限を指定

https://github.com/akokubo/retro_shooting/tree/59bea39078be6699da47022eda26530f297d8ff3/retro_shooting

Cannon タブ

```
// キャノン・クラス
class Cannon extends Sprite {
    float speed; // 速度

    // コンストラクタ
    Cannon(PImage image, float x, float y) {
        super(image, x, y);

        // 速度の指定
        speed = 2;
    }

    // 移動(オーバーライド)
    void move() {
        x -= mouseX;
        // マウスの方向に speed 分移動する
        if (x < mouseX) {
            x += speed;
        } else if (x > mouseX) {
            x -= speed;
        }
    }
}
```

メイン、Sprite、Stage タブ

変更なし

7. キャノンがレールからはみ出さないようにする

https://github.com/akokubo/retro_shooting/tree/e531d976907c80ac93d5a09603aeb176c7b73ed4/retro_shooting

Cannon タブ

```
// キャノン・クラス
class Cannon extends Sprite {
[中略]

// 移動(オーバーライド)
void move() {
    // マウスの方向に speed 分移動する
if (x < mouseX) {
    if (x < mouseX && x < stage.image.width - this.image.width / 2) {
        x += speed;
} else if (x > mouseX) {
    } else if (x > mouseX && x > this.image.width / 2) {
        x -= speed;
    }
}
}
```

メイン、Sprite、Stage タブ

変更なし

8. レーザー・クラスの実装

https://github.com/akokubo/retro_shooting/tree/07f376901278bc489f43db108b616deb09e37428/retro_shooting

Laser タブ

```
// レーザー・クラス
class Laser extends Sprite {
    float speed; // 速度

    // コンストラクタ
    Laser(PImage image, float x, float y) {
        super(image);

        // 座標の設定
        this.x = x;
        this.y = y;

        // 速度の指定
        speed = 4;
    }

    // 移動(オーバーライド)
    void move() {
        y -= speed;
    }
}
```

メイン・タブ

```
// オブジェクト
Stage stage;
Cannon cannon;
Laser laser;

void setup() {
    [中略]
}

void draw() {
    // 残像を消す
    background(204);

    // キャノンの移動
    cannon.move();

    // レーザーの発射
    if (laser == null) {
        // レーザーがないときにマウスをクリックしたら
        if (mousePressed) {
            // レーザーをキャノンの位置に生成
            laser = new Laser(loadImage("laser.png"), cannon.x, cannon.y - 8);
        }
    } else {
        // レーザーがあるときは、移動させる
        laser.move();
    }
}
```

```
// 表示
stage.display();
cannon.display();
if (laser != null) {
    // レーザーが存在するとき、表示させる
    laser.display();
}
}
```

Cannon、Sprite、Stage タブ

変更なし

9. レーザーが画面外に出たら消す

https://github.com/akokubo/retro_shooting/tree/d1c3e61dad963142f080dcdff6b35212d6d5908f/retro_shooting

Laser タブ

```
// レーザー・クラス
class Laser extends Sprite {
[中略]

// 移動(オーバーライド)
void move() {
    y -= speed;
}

// 画面外に出たか
boolean isOver() {
    boolean result = false;

    if (y < 0) {
        result = true;
    }

    return result;
}
}
```

メイン・タブ

```
// オブジェクト
[中略]

void setup() {
[中略]
}

void draw() {
[中略]

// レーザーの発射
if (laser == null) {
    // レーザーがないときにマウスをクリックしたら
    if (mousePressed) {
        // レーザーをキャノンの位置に生成
        laser = new Laser(loadImage("laser.png"), cannon.x, cannon.y - 8);
    }
} else {
    // レーザーがあるときは、移動させる
    laser.move();
    // レーザーが画面外に出たら
    if (laser.isOver()) {
        // レーザーを消す
        laser = null;
    }
}

// 表示
```

[中略]

}

Cannon、Sprite、Stage タブ

変更なし

10.エイリアン・クラスの作成

https://github.com/akokubo/retro_shooting/tree/86c2ffbf3b32ab00d1732ab6697a09307afb98f8/retro_shooting

Alien タブ

```
// エイリアン・クラス
class Alien extends Sprite {
  int direction; // 移動方向(左-1、右+1)
  float speed; // 速度
  float advance; // 前進速度

  // コンストラクタ
  Alien(PImage image, float x, float y) {
    super(image, x, y);

    // 移動方向と速度の設定
    direction = 1;
    speed = 0.5;
    advance = 8;
  }

  // 移動(オーバーライド)
  void move() {
    x += direction * speed;
    // 画面の端に着いたら
    if (atEnds()) {
      // ターンする
      turn();
    }
  }

  // 画面の端に着いた
  boolean atEnds() {
    boolean result = false;

    if (x < this.image.width / 2 || x > stage.image.width - this.image.width / 2) {
      result = true;
    }

    return result;
  }

  // ターンする
  void turn() {
    // 向きを変えて
    direction *= -1;
    // 下がる
    y += advance;
  }
}
```

メイン・タブ

```
// オブジェクト
Stage stage;
Cannon cannon;
```

```

Laser laser;
Alien alien;

void setup() {
[中略]

    // オブジェクトの生成
    stage = new Stage(loadImage("stage.png"));
    cannon = new Cannon(loadImage("cannon.png"),
        stage.image.width / 2, stage.image.height - 38);
    alien = new Alien(loadImage("alien.png"), 20, 20);
}

void draw() {
[中略]

    // レーザーの発射
    if (laser == null) {
        // レーザーがないときにマウスをクリックしたら
        if (mousePressed) {
            // レーザーをキャノンの位置に生成
            laser = new Laser(loadImage("laser.png"), cannon.x, cannon.y - 8);
        }
    } else {
        // レーザーがあるときは、移動させる
        laser.move();

        // レーザーが画面外に出たら
        if (laser.isOver()) {
            // レーザーを消す
            laser = null;
        }
    }

    // エイリアンの移動
    alien.move();

    // 表示
    stage.display();
    cannon.display();
    if (laser != null) {
        // レーザーが存在するとき、表示させる
        laser.display();
    }
    alien.display();
}

```

Cannon、Laser、Sprite、Stage タブ

変更なし

11.エイリアンを増やす

https://github.com/akokubo/retro_shooting/tree/abb265e753400922cf104073a4974ea47e662a60/retro_shooting

メイン・タブ

```
// オブジェクト
Stage stage;
Cannon cannon;
Laser laser;
Alien alien;
ArrayList<Alien> aliens;

void setup() {
  [中略]

  // オブジェクトの生成
  stage = new Stage(loadImage("stage.png"));
  cannon = new Cannon(loadImage("cannon.png"),
    stage.image.width / 2, stage.image.height - 38);
alien = new Alien(loadImage("alien.png"), 20, 20);

  // エイリアン群の生成
  aliens = new ArrayList<Alien>();
  // 一つ一つのエイリアンを生成
  for (int y = 20; y <= 120; y += 20) {
    for (int x = 20; x <= 340; x+= 32) {
      // エイリアンを生成し
      Alien alien = new Alien(loadImage("alien.png"), x, y);
      // エイリアン群に追加
      aliens.add(alien);
    }
  }
}

void draw() {
  [中略]

// エイリアンの移動
alien.move();
  // エイリアン群の移動
  for (int i = 0; i < aliens.size(); i++) {
    // エイリアンを1つ取り出し
    Alien alien = aliens.get(i);
    // エイリアンを移動させる
    alien.move();
  }

  // 表示
  [中略]
alien.display();
  for (int i = 0; i < aliens.size(); i++) {
    // エイリアンを一つ取り出し
    Alien alien = aliens.get(i);
    // エイリアンを表示する
    alien.display();
  }
}
```

}

Alien、Cannon、Laser、Sprite、Stage タブ

変更なし

12.エイリアンが画面端に着いたら、全エイリアンをターンさせる

https://github.com/akokubo/retro_shooting/tree/c0b293741ce5954444c5790cb0a95cb126aa4715/retro_shooting

Alien タブ

```
// エイリアン・クラス
class Alien extends Sprite {
[中略]

// 移動(オーバーライド)
void move() {
    x += direction * speed;
    // 画面の端に着いたら
    if (atEnds()) {
        // ターンする
        turn();
        // 全エイリアンにターンを指示する
        for (int i = 0; i < aliens.size(); i++) {
            Alien alien = aliens.get(i);
            alien.turn();
        }
    }
}
[中略]
}
```

メイン、Cannon、Laser、Sprite、Stage タブ

変更なし

13. レーザーとエイリアンの当たり判定の作成

https://github.com/akokubo/retro_shooting/tree/862fcd91b2dc6f0e6c31b2c9a45c1f1087dea00f/retro_shooting

メイン・タブ

```
[中略]

void setup() {
[中略]
}

void draw() {
[中略]

    // エイリアン群の移動
    for (int i = 0; i < aliens.size(); i++) {
        // エイリアンを1つ取り出し
        Alien alien = aliens.get(i);
        // エイリアンを移動させる
        alien.move();
    }

    // レーザーとエイリアンの当たり判定
    for (int i = 0; i < aliens.size(); i++) {
        // エイリアンを一つ取り出し
        Alien alien = aliens.get(i);
        // レーザーが存在して、そのレーザーがエイリアンに当たったら
        if (laser != null && laser.isContactedWith(alien)) {
            // レーザーを消して
            laser = null;

            // エイリアンを消す
            aliens.remove(alien);
        }
    }

    // 表示
[中略]
}
```

Alien、Cannon、Laser、Sprite、Stage タブ

変更なし

14. スコアの作成

https://github.com/akokubo/retro_shooting/tree/acc6685746c9d50efc50e8031f017ae9d886198c/retro_shooting

Stage タブ

```
// ステージ・クラス
class Stage extends Sprite {
    PFont font; // フォント
    int score; // スコア

    // コンストラクタ
    Stage(PImage image) {
        super(image);

        // 画像の中心を座標として設定
        x = image.width / 2;
        y = image.height / 2;

        // フォントの設定
        font = createFont("MS Gothic", 20);

        // スコアの初期化
        score = 0;
    }

    // 表示(オーバーライド)
    void display() {
        // 親クラスのメソッドをそのまま呼ぶ
        super.display();

        // フォントと色の設定
        textFont(font);
        fill(0);

        // スコアの表示(仮)
        text("SCORE: 14000", this.image.width + 20, 20);
        // スコアの表示
        text("SCORE: " + score, this.image.width + 20, 20);

        // ライフの表示(仮)
        text("LIFE: 3", this.image.width + 20, 40);
    }

    // スコアアップ
    void scoreUp(int value) {
        score += value;
    }
}
```

メイン・タブ

[中略]

```
void setup() {
```

[中略]

```
}
```

```
void draw() {
```

[中略]

```
// レーザーとエイリアンの当たり判定
```

```
for (int i = 0; i < aliens.size(); i++) {
```

```
    // エイリアンを一つ取り出し
```

```
    Alien alien = aliens.get(i);
```

```
    // レーザーが存在して、そのレーザーがエイリアンに当たったら
```

```
    if (laser != null && laser.isContactedWith(alien)) {
```

```
        // レーザーを消して
```

```
        laser = null;
```

```
        // エイリアンを消す
```

```
        aliens.remove(alien);
```

```
        // スコアをアップする
```

```
        stage.scoreUp(100);
```

```
    }
```

```
}
```

[中略]

```
}
```

Alien、Cannon、Laser、Sprite タブ

変更なし

15. キャノンとエイリアンの当たり判定とライフの作成

https://github.com/akokubo/retro_shooting/tree/c91afb7d54abb0fa2db7dc91cbf6e63a02d27abb/retro_shooting

Cannon タブ

```
// キャノン・クラス
class Cannon extends Sprite {
    float speed; // 速度
    int life; // ライフ

    // コンストラクタ
    Cannon(PImage image, float x, float y) {
        super(image, x, y);

        // 速度の指定
        speed = 2;

        // ライフの初期値
        life = 3;
    }

    // 移動(オーバーライド)
    void move() {
        if (x < mouseX && x < stage.image.width - this.image.width / 2) {
            x += speed;
        } else if (x > mouseX && x > this.image.width / 2) {
            x -= speed;
        }
    }

    // キャノンの破壊
    void destroy() {
        life--;
    }
}
```

Stage タブ

```
// ステージ・クラス
class Stage extends Sprite {
[中略]

    // 表示(オーバーライド)
    void display() {
        // 親クラスのメソッドをそのまま呼ぶ
        super.display();

        // フォントと色の設定
        textFont(font);
        fill(0);

        // スコアの表示
        text("SCORE: " + score, this.image.width + 20, 20);

        // ライフの表示(仮)
        text("LIFE: 3", this.image.width + 20, 40);
    }
}
```

```

// ライフの表示
text("LIFE: " + cannon.life, this.image.width + 20, 40);
}

// スコアアップ
void scoreUp(int value) {
    score += value;
}
}

```

メイン・タブ

```

[中略]

void setup() {
[中略]
}

void draw() {
[中略]
// レーザーとエイリアンの当たり判定
for (int i = 0; i < aliens.size(); i++) {
    // エイリアンを一つ取り出し
    Alien alien = aliens.get(i);
    // レーザーが存在して、そのレーザーがエイリアンに当たったら
    if (laser != null && laser.isContactedWith(alien)) {
        // レーザーを消して
        laser = null;

        // エイリアンを消す
        aliens.remove(alien);

        // スコアをアップする
        stage.scoreUp(100);
    }
}

// キャノンとエイリアンの当たり判定
for (int i = 0; i < aliens.size(); i++) {
    Alien alien = aliens.get(i);
    // キャノンがエイリアンに当たったら
    if (cannon.isContactedWith(alien)) {
        // キャノンを破壊して
        cannon.destroy();
        // エイリアンを消す
        aliens.remove(alien);
    }
}

// 表示
[中略]
}

```

※実際には大幅なインデントの変更がある

Alien、Laser、Sprite タブ

変更なし

16. ゲームオーバーの作成

https://github.com/akokubo/retro_shooting/tree/0e7242a120df522ca71b166f7d50ec06a7660acb/retro_shooting

Stage タブ

```
// ステージ・クラス
class Stage extends Sprite {
[中略]

    // 表示(オーバーライド)
    void display() {
        // 親クラスのメソッドをそのまま呼ぶ
        super.display();

        // フォントと色の設定
        textFont(font);
        fill(0);

        // スコアの表示
        text("SCORE: " + score, this.image.width + 20, 20);

        // ライフの表示
        text("LIFE: " + cannon.life, this.image.width + 20, 40);

        // ゲームオーバーの表示
        if (isGameOver()) {
            fill(255, 0, 0);
            text("Game Over", this.image.width / 2 - 50, this.image.height / 2 - 10);
        }
    }

    // スコアアップ
    void scoreUp(int value) {
        score += value;
    }

    // ゲームオーバーか
    boolean isGameOver() {
        boolean result = false;

        // ライフがゼロ
        if (cannon.life <= 0) {
            result = true;
        }

        return result;
    }
}
```

メイン・タブ

```
[中略]

void setup() {
[中略]
}
```

```

void draw() {
    // 残像を消す
    background(204);

    // ゲームオーバーになっていなければ
    if (!stage.isGameOver()) {
        // キャノンの移動
        cannon.move();

        [中略]

        // キャノンとエイリアンの当たり判定
        for (int i = 0; i < aliens.size(); i++) {
            Alien alien = aliens.get(i);
            // キャノンがエイリアンに当たったら
            if (cannon.isContactedWith(alien)) {
                // キャノンを破壊して
                cannon.destroy();
                // エイリアンを消す
                aliens.remove(alien);
            }
        }
    }

    [中略]
}

```

※実際には大幅なインデントの変更がある

Alien、Cannon、Laser、Sprite タブ

変更なし

17. 勝利の作成

https://github.com/akokubo/retro_shooting/tree/f61ddbf4164e840bd74ddbf61db87f274207ac88/retro_shooting

Stage タブ

```
// ステージ・クラス
class Stage extends Sprite {
[中略]

    // 表示(オーバーライド)
    void display() {
[中略]

        // ゲームオーバーの表示
        if (isGameOver()) {
            fill(255, 0, 0);
            text("Game Over", this.image.width / 2 - 50, this.image.height / 2 - 10);
        }

        // 勝利の表示
        if (isWin()) {
            fill(255, 0, 0);
            text("You Win", this.image.width / 2 - 50, this.image.height / 2 - 10);
        }
    }

    // スコアアップ
    void scoreUp(int value) {
        score += value;
    }

    // ゲームオーバーか
    boolean isGameOver() {
        boolean result = false;

        // ライフがゼロ
        if (cannon.life <= 0) {
            result = true;
        }

        return result;
    }

    // ゲームに勝ったか
    boolean isWin() {
        boolean result = false;

        // エイリアンをすべて倒した
        if (aliens.size() == 0) {
            result = true;
        }

        return result;
    }
}
```

メイン・タブ

[中略]

```
void setup() {
```

[中略]

```
}
```

```
void draw() {
```

```
  // 残像を消す
```

```
  background(204);
```

```
  // ゲームオーバーになっていなければ
```

```
  if (!stage.isGameOver()) {
```

```
  // ゲームが終了していなければ
```

```
  if (!stage.isGameOver() && !stage.isWin()) {
```

```
    // キャノンの移動
```

```
    cannon.move();
```

[中略]

```
}
```

Alien、Cannon、Laser、Sprite タブ

変更なし

18. 爆弾クラスの作成

https://github.com/akokubo/retro_shooting/tree/4052866f599048feb8da78737a1b4e4ebfc3faa9/retro_shooting

Bomb タブ

```
// 爆弾クラス
class Bomb extends Sprite {
    float speed; // 速度

    // コンストラクタ
    Bomb(PImage image, float x, float y) {
        super(image, x, y);

        // 速度の指定
        speed = 1;
    }

    // 移動(オーバーライド)
    void move() {
        y += speed;
    }

    // 画面外に出たか
    boolean isOver() {
        boolean result = false;

        if (y >= stage.image.height) {
            result = true;
        }

        return result;
    }
}
```

メイン・タブ

```
// オブジェクト
Stage stage;
Cannon cannon;
Laser laser;
ArrayList<Alien> aliens;
ArrayList<Bomb> bombs;

void setup() {
[中略]

    // エイリアン群の生成
    aliens = new ArrayList<Alien>();
    // 一つ一つのエイリアンの生成
    for (int y = 20; y <= 120; y += 20) {
        for (int x = 20; x <= 340; x+= 32) {
            // エイリアンを生成し
            Alien alien = new Alien(loadImage("alien.png"), x, y);
            // エイリアン群に追加
            aliens.add(alien);
        }
    }
}
```

```

// 爆弾群の生成
bombs = new ArrayList<Bomb>();
}

void draw() {
[中略]

// エイリアン群の移動
for (int i = 0; i < aliens.size(); i++) {
    // エイリアンを1つ取り出し
    Alien alien = aliens.get(i);
    // エイリアンを移動させる
    alien.move();
}

// 爆弾の投下
for (int i = 0; i < aliens.size(); i++) {
    // エイリアンを1つ取り出し
    Alien alien = aliens.get(i);
    // ある確率(60 秒に 1 回)で
    if (random(60 * frameRate) < 1) {
        // 爆弾を生成
        Bomb bomb = new Bomb(loadImage("bomb.png"), alien.x, alien.y);
        // 爆弾群に追加
        bombs.add(bomb);
    }
}

// 爆弾群の移動
for (int i = 0; i < bombs.size(); i++) {
    // 爆弾を1つ取り出し
    Bomb bomb = bombs.get(i);
    // 爆弾を移動させる
    bomb.move();
    // 爆弾が画面外に出たら
    if (bomb.isOver()) {
        // 爆弾を消す
        bombs.remove(bomb);
    }
}

[中略]
for (int i = 0; i < aliens.size(); i++) {
    // エイリアンを一つ取り出し
    Alien alien = aliens.get(i);
    // エイリアンを表示する
    alien.display();
}
for (int i = 0; i < bombs.size(); i++) {
    // 爆弾を一つ取り出し
    Bomb bomb = bombs.get(i);
    // 爆弾を表示する
    bomb.display();
}
}

```

Alien、Cannon、Laser、Sprite、Stage タブ

変更なし

19. キャノンと爆弾の当たり判定の作成

https://github.com/akokubo/retro_shooting/tree/f19d88138091efcff4c3f62f50d42eb6f23dc44a/retro_shooting

メイン・タブ

[中略]

```
void setup() {
```

[中略]

```
}
```

```
void draw() {
```

[中略]

```
    // キャノンとエイリアンの当たり判定
    for (int i = 0; i < aliens.size(); i++) {
        Alien alien = aliens.get(i);
        // キャノンがエイリアンに当たったら
        if (cannon.isContactedWith(alien)) {
            // キャノンを破壊して
            cannon.destroy();
            // エイリアンを消す
            aliens.remove(alien);
        }
    }
}
```

```
    // キャノンと爆弾の当たり判定
    for (int i = 0; i < bombs.size(); i++) {
        Bomb bomb = bombs.get(i);
        // キャノンが爆弾に当たったら
        if (cannon.isContactedWith(bomb)) {
            // キャノンを破壊して
            cannon.destroy();
            // 爆弾を消す
            bombs.remove(bomb);
        }
    }
}
```

```
    // 表示
```

[中略]

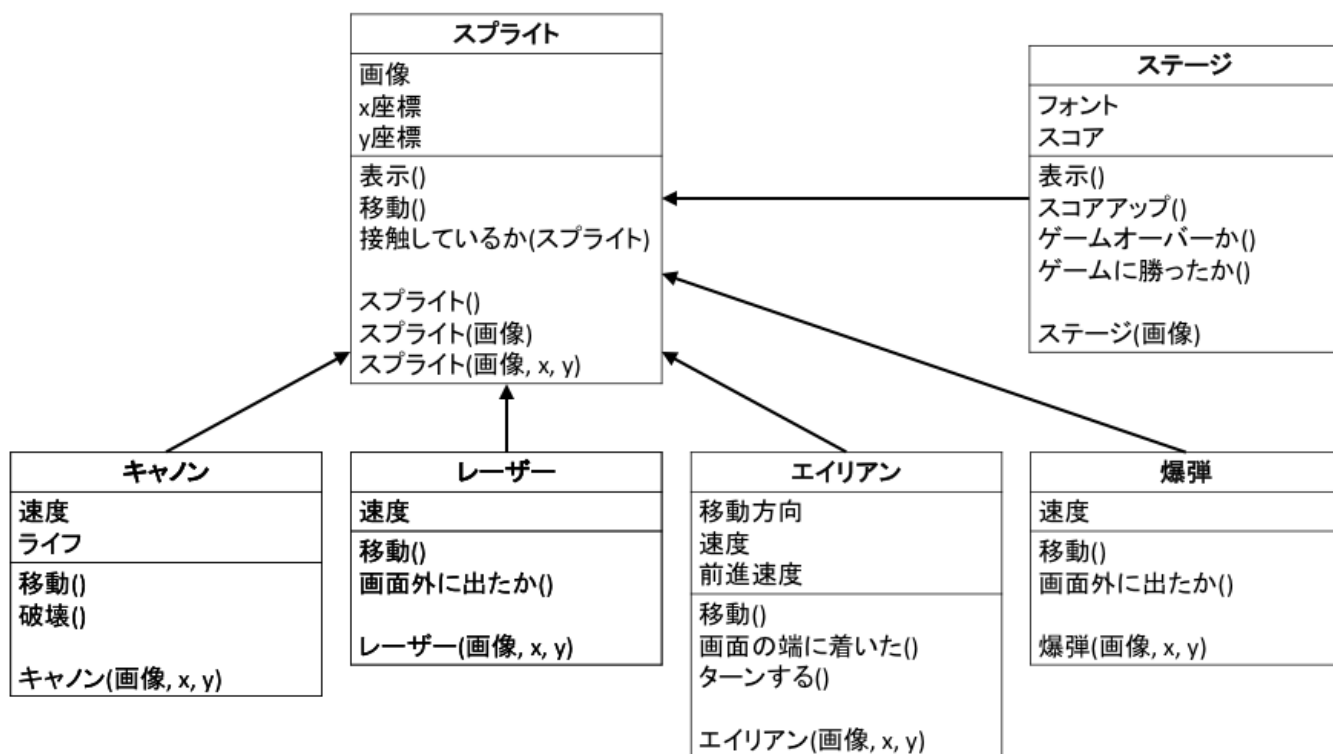
```
}
```

Alien、Bomb、Cannon、Laser、Sprite、Stage タブ

変更なし

20. プログラムの最終版

https://github.com/akokubo/retro_shooting/retro_shooting



メイン
ステージ
キャノン
レーザー
エイリアン群
爆弾群
初期化()
描画()

```

初期化() {
    ステージの生成
    キャノンの生成
    エイリアン群の生成
    爆弾群の生成
}
    
```

```

描画() {
    残像を消す
    if(ゲームが終了していない) {
        キャノンの移動
        レーザーの発射と移動
        エイリアン群の移動
        爆弾の投下
        爆弾群の移動
        レーザーとエイリアンの当たり判定
        キャノンとエイリアンの当たり判定
        キャノンと爆弾の当たり判定
    }
    ステージを表示
    キャノンを表示
    レーザーを表示
    エイリアン群を表示
    爆弾群を表示
}
    
```

メイン・タブ

```
// オブジェクト
Stage stage;
Cannon cannon;
Laser laser;
ArrayList<Alien> aliens;
ArrayList<Bomb> bombs;

void setup() {
    // ディスプレイ・ウィンドウのサイズを 640x360 に
    size(640, 360);

    // 画像を中心に表示するモード
    imageMode(CENTER);

    // オブジェクトの生成
    stage = new Stage(loadImage("stage.png"));
    cannon = new Cannon(loadImage("cannon.png"),
        stage.image.width / 2, stage.image.height - 38);

    // エイリアン群の生成
    aliens = new ArrayList<Alien>();
    // 一つ一つのエイリアンを生成
    for (int y = 20; y <= 120; y += 20) {
        for (int x = 20; x <= 340; x+= 32) {
            // エイリアンを生成し
            Alien alien = new Alien(loadImage("alien.png"), x, y);
            // エイリアン群に追加
            aliens.add(alien);
        }
    }

    // 爆弾群の生成
    bombs = new ArrayList<Bomb>();
}

void draw() {
    // 残像を消す
    background(204);

    // ゲームが終了していなければ
    if (!stage.isGameOver() && !stage.isWin()) {
        // キャノンの移動
        cannon.move();

        // レーザーの発射
        if (laser == null) {
            // レーザーがないときにマウスをクリックしたら
            if (mousePressed) {
                // レーザーをキャノンの位置に生成
                laser = new Laser(loadImage("laser.png"), cannon.x, cannon.y - 8);
            }
        } else {
            // レーザーがあるときは、移動させる
            laser.move();

            // レーザーが画面外に出たら
```



```

    if (laser.isOver()) {
        // レーザーを消す
        laser = null;
    }
}

// エイリアン群の移動
for (int i = 0; i < aliens.size(); i++) {
    // エイリアンを1つ取り出し
    Alien alien = aliens.get(i);
    // エイリアンを移動させる
    alien.move();
}

// 爆弾の投下
for (int i = 0; i < aliens.size(); i++) {
    // エイリアンを1つ取り出し
    Alien alien = aliens.get(i);
    // ある確率(60 秒に 1 回)で
    if (random(60 * frameRate) < 1) {
        // 爆弾を生成
        Bomb bomb = new Bomb(loadImage("bomb.png"), alien.x, alien.y);
        // 爆弾群に追加
        bombs.add(bomb);
    }
}

// 爆弾群の移動
for (int i = 0; i < bombs.size(); i++) {
    // 爆弾を1つ取り出し
    Bomb bomb = bombs.get(i);
    // 爆弾を移動させる
    bomb.move();
    // 爆弾が画面外に出たら
    if (bomb.isOver()) {
        // 爆弾を消す
        bombs.remove(bomb);
    }
}

// レーザーとエイリアンの当たり判定
for (int i = 0; i < aliens.size(); i++) {
    // エイリアンを一つ取り出し
    Alien alien = aliens.get(i);
    // レーザーが存在して、そのレーザーがエイリアンに当たったら
    if (laser != null && laser.isContactedWith(alien)) {
        // レーザーを消して
        laser = null;

        // エイリアンを消す
        aliens.remove(alien);

        // スコアをアップする
        stage.scoreUp(100);
    }
}

// キャノンとエイリアンの当たり判定

```

```

    for (int i = 0; i < aliens.size(); i++) {
        Alien alien = aliens.get(i);
        // キャノンがエイリアンに当たったら
        if (cannon.isContactedWith(alien)) {
            // キャノンを破壊して
            cannon.destroy();
            // エイリアンを消す
            aliens.remove(alien);
        }
    }
}

// キャノンと爆弾の当たり判定
for (int i = 0; i < bombs.size(); i++) {
    Bomb bomb = bombs.get(i);
    // キャノンが爆弾に当たったら
    if (cannon.isContactedWith(bomb)) {
        // キャノンを破壊して
        cannon.destroy();
        // 爆弾を消す
        bombs.remove(bomb);
    }
}

// 表示
stage.display();
cannon.display();
if (laser != null) {
    // レーザーが存在するとき、表示させる
    laser.display();
}
for (int i = 0; i < aliens.size(); i++) {
    // エイリアンを一つ取り出し
    Alien alien = aliens.get(i);
    // エイリアンを表示する
    alien.display();
}
for (int i = 0; i < bombs.size(); i++) {
    // 爆弾を一つ取り出し
    Bomb bomb = bombs.get(i);
    // 爆弾を表示する
    bomb.display();
}
}

```

Sprite タブ

```

// スプライト・クラス
class Sprite {
    // 画像
    PImage image;

    // 座標
    float x;
    float y;

    // コンストラクタ(デフォルト)
    Sprite() {

```

```

}

// コンストラクタ (画像を指定するとき)
Sprite(PImage image) {
    this.image = image;
}

// コンストラクタ (画像と座標を指定するとき)
Sprite(PImage image, float x, float y) {
    this.image = image;
    this.x = x;
    this.y = y;
}

// 表示
void display() {
    image(this.image, x, y);
}

// 移動
void move() {
}

// 当たり判定
boolean isContactedWith(Sprite sprite) {
    // 判定結果を入れる変数。デフォルトは false
    boolean result = false;

    // 当たったら true に
    if (dist(x, y, sprite.x, sprite.y) < (image.width + sprite.image.width) / 2) {
        result = true;
    }

    return result;
}
}

```

Stage タブ

```

// ステージ・クラス
class Stage extends Sprite {
    PFont font; // フォント
    int score; // スコア

    // コンストラクタ
    Stage(PImage image) {
        super(image);

        // 画像の中心を座標として設定
        x = image.width / 2;
        y = image.height / 2;

        // フォントの生成
        font = createFont("MS Gothic", 20);

        // スコアの初期化
        score = 0;
    }
}

```

```

// 表示(オーバーライド)
void display() {
    // 親クラスのメソッドをそのまま呼ぶ
    super.display();

    // フォントと色の設定
    textFont(font);
    fill(0);

    // スコアの表示
    text("SCORE: " + score, this.image.width + 20, 20);

    // ライフの表示
    text("LIFE: " + cannon.life, this.image.width + 20, 40);

    // ゲームオーバーの表示
    if (isGameOver()) {
        fill(255, 0, 0);
        text("Game Over", this.image.width / 2 - 50, this.image.height / 2 - 10);
    }

    // 勝利の表示
    if (isWin()) {
        fill(255, 0, 0);
        text("You Win", this.image.width / 2 - 50, this.image.height / 2 - 10);
    }
}

// スコアアップ
void scoreUp(int value) {
    score += value;
}

// ゲームオーバーか
boolean isGameOver() {
    boolean result = false;

    // ライフがゼロ
    if (cannon.life <= 0) {
        result = true;
    }

    return result;
}

// ゲームに勝ったか
boolean isWin() {
    boolean result = false;

    // エイリアンをすべて倒した
    if (aliens.size() == 0) {
        result = true;
    }

    return result;
}
}

```

Cannon タブ

```
// キャノン・クラス
class Cannon extends Sprite {
    float speed; // 速度
    int life; // ライフ

    // コンストラクタ
    Cannon(PImage image, float x, float y) {
        super(image, x, y);

        // 速度の指定
        speed = 2;

        // ライフの初期値
        life = 3;
    }

    // 移動(オーバーライド)
    void move() {
        // マウスの方向に speed 分移動する
        if (x < mouseX && x < stage.image.width - this.image.width / 2) {
            x += speed;
        } else if (x > mouseX && x > this.image.width / 2) {
            x -= speed;
        }
    }

    // キャノンの破壊
    void destroy() {
        life--;
    }
}
```

Laser タブ

```
// レーザー・クラス
class Laser extends Sprite {
    float speed; // 速度

    // コンストラクタ
    Laser(PImage image, float x, float y) {
        super(image);

        // 座標の設定
        this.x = x;
        this.y = y;

        // 速度の指定
        speed = 4;
    }

    // 移動(オーバーライド)
    void move() {
        y -= speed;
    }
}
```

```

// 画面外に出たか
boolean isOver() {
    boolean result = false;

    if (y < 0) {
        result = true;
    }

    return result;
}
}

```

Alien タブ

```

// エイリアン・クラス
class Alien extends Sprite {
    int direction; // 移動方向(左-1、右+1)
    float speed; // 速度
    float advance; // 前進速度

    // コンストラクタ
    Alien(PImage image, float x, float y) {
        super(image, x, y);

        // 移動方向と速度の設定
        direction = 1;
        speed = 0.5;
        advance = 8;
    }

    // 移動(オーバーライド)
    void move() {
        x += direction * speed;
        // 画面の端に着いたら
        if (atEnds()) {
            // 全エイリアンにターンを指示する
            for (int i = 0; i < aliens.size(); i++) {
                Alien alien = aliens.get(i);
                alien.turn();
            }
        }
    }

    // 画面の端に着いた
    boolean atEnds() {
        boolean result = false;

        if (x < this.image.width / 2 || x > stage.image.width - this.image.width / 2) {
            result = true;
        }

        return result;
    }

    // ターンする
    void turn() {
        // 向きを変えて
        direction *= -1;
    }
}

```

```
    // 下がる  
    y += advance;  
}  
}
```

Bomb タブ

```
// 爆弾クラス  
class Bomb extends Sprite {  
    float speed; // 速度  
  
    // コンストラクタ  
    Bomb(PImage image, float x, float y) {  
        super(image, x, y);  
  
        // 速度の指定  
        speed = 1;  
    }  
  
    // 移動(オーバーライド)  
    void move() {  
        y += speed;  
    }  
  
    // 画面外に出たか  
    boolean isOver() {  
        boolean result = false;  
  
        if (y >= stage.image.height) {  
            result = true;  
        }  
  
        return result;  
    }  
}
```