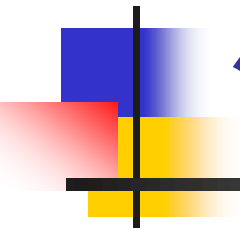
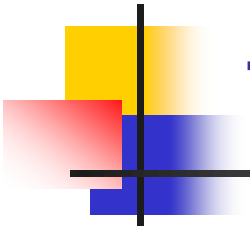


ゲームをつくる: 2
シューティングゲーム（2）

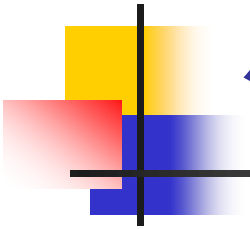
プログラミング演習I 第13回

2016/06/03

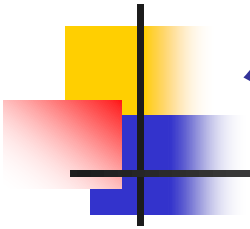


予定

- 5/30(月)
  - シューティングゲームをつくる（1）
- 6/3(金)
  - シューティングゲームをつくる（2）

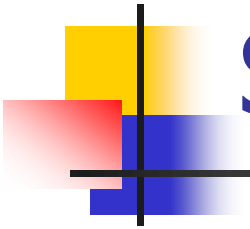


ゲームらしくする



ゲーム要素の導入

- Step 5: 敵を動かす
- Step 6: 勝敗を決める
- Step 7: ゲーム性を高める



Step 5: 敵を動かす

- 敵機の移動
- 自機との衝突

shooting10

Step 5-1: 敵機の移動

敵機パラメータ部分

```
// 敵機のパラメータ
int eN = 3;  // 残機数
int eS = 30;   // サイズ
int eX = 200;  // X座標
int eY = 50;   // Y座標
int eV = 2;    // 速度
```

敵機の速度パラメータを追加

敵機処理部分

```
// 敵機
 fill(255);
 rect(eX, eY, eS, eS);
 eX += eV;
```

横に移動

敵機処理部分

```
// 敵機
 fill(255);
 rect(eX, eY, eS, eS);
 eX += eV;
 if (eX > width || eX < 0) {
   eV *= -1;
 }
```

端で折り返し

敵機処理部分

```
// 敵機
 fill(255);
 rect(eX, eY, eS, eS);
 eX += eV;
 if (eX > width || eX < 0) {
   eV *= -1;
   eY += eS;
 }
```

折り返す時
縦にも移動

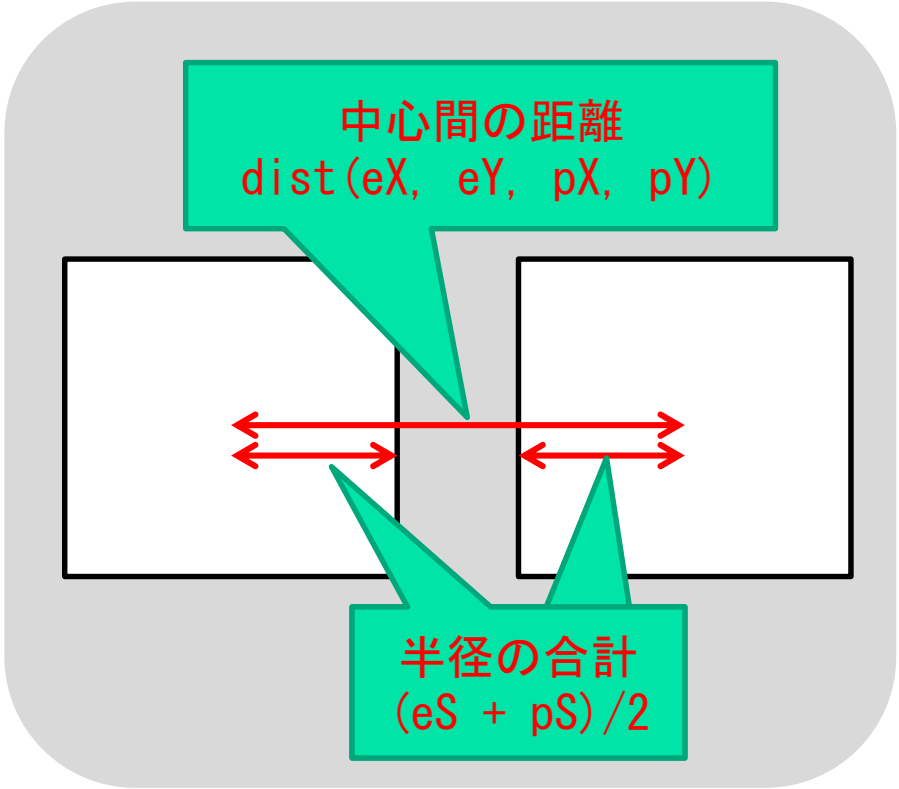
Step 5-2: 自機との衝突

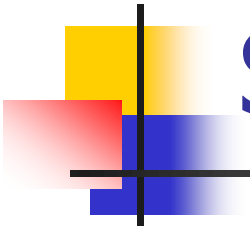
- 敵機と自機の衝突判定

自機処理部分

```
// 自機
 fill(255);
 rect(pX, pY, pS, pS);
 if (dist(eX, eY, pX, pY) < (eS + pS) / 2) {
   scene = 2;
 }
```

衝突したら
シーン2(結果画面)に遷移





Step 6: 勝敗を決める

- ルールを決める
- 勝敗の表示

Step 6-1: ルールの設定

- 自分の勝ち
  - 敵機に衝突されずに3機撃墜
- 敵の勝ち
  - 3機撃墜される前に自機に衝突

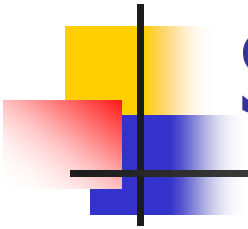
Step 6-2: 勝敗判定と表示

Resultタブ

```
// 結果画面

void result() {
  background(0);   // 背景を黒く
  fill(255);       // 文字を白く

  // 勝敗判定
  if (eN == 0) {
    text("あなたの勝ちです (^_^)", 0, 100);
  }
  else {
    text("あなたの負けです (*_*)", 0, 100);
  }

  // マウス入力
  if (mousePressed) {
    scene = 3;  // シーン切り替え
  }
}
```

残機数が0なら勝ち

そうでなければ負け



Step 7: ゲーム性を高める

- パラメータの調整
- （例１）敵の動きを変える
- （例2）敵が反撃する
- 高機能化の例
- 装飾の例

Step 7-1: 難易度を調整する

- パラメータ調整
  - 敵のパラメータ
    - 大きさ、残機数、速さ
  - 自機のパラメータ
    - 速さ
  - 自弾のパラメータ
    - 速さ
  - 全体のパラメータ
    - ステージの大きさ
- その他の数値
  - 敵の前進の仕方
  - 敵の加速の仕方

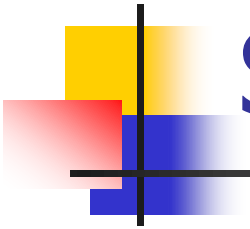
Step 7-2:（例1）敵の動きを変える

自弾処理部分

```
// 自弾
if (psL) {
  fill(255);
  ellipse(psX, psY, psS, psS);
  psY -= psV;
  if (psY < 0) {
    psL = false;
  }
  if (dist(eX, eY, psX, psY) < (eS + psS) / 2) {
    background(255, 0, 0);
    eN -= 1;
    eV *= 2;
    psL = false;
    if (eN == 0) {
      scene = 2;
    }
  }
}
```

自弾の当たり判定の部分に追加

当たったら速度を倍に

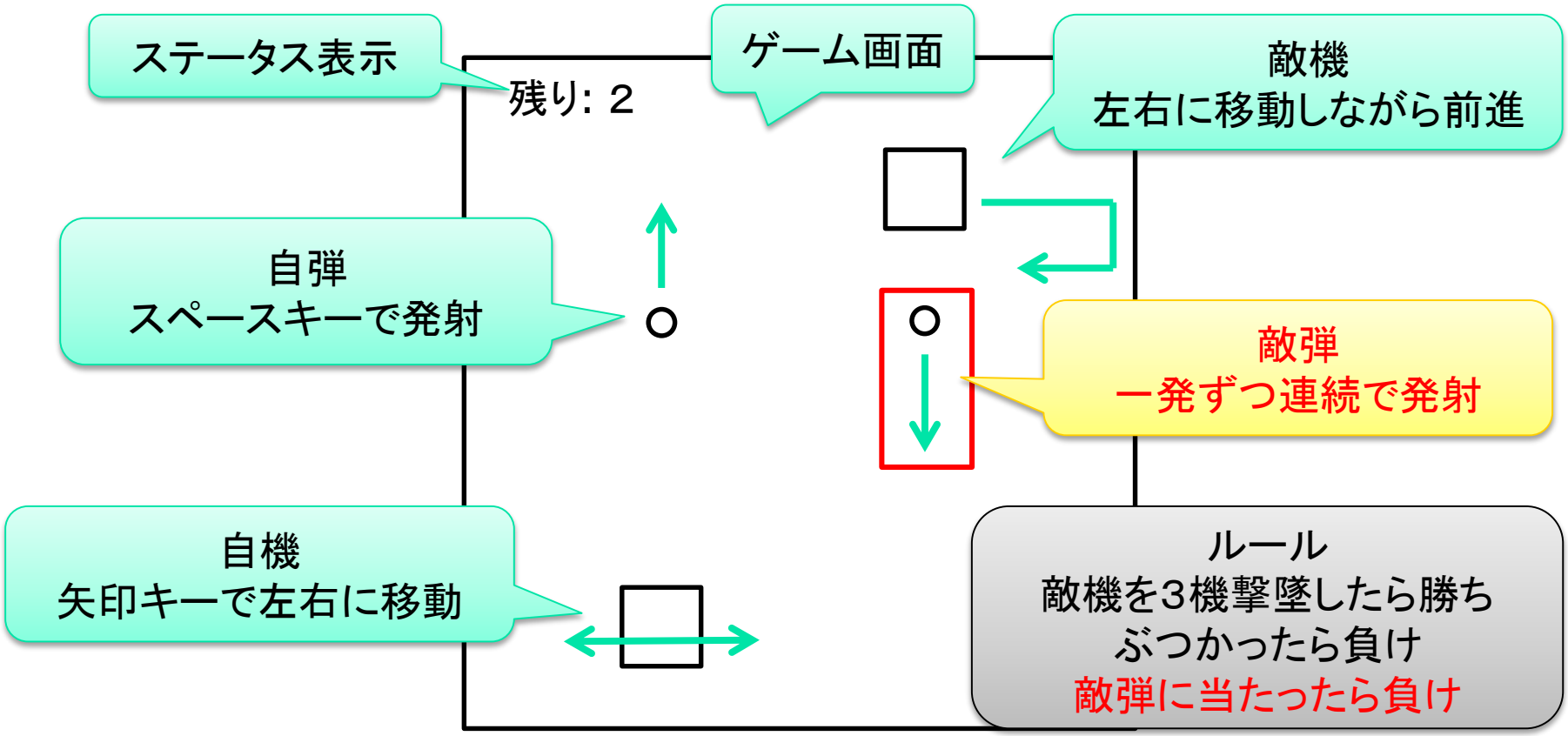


Step 7-3:（例2）敵が反撃する

- 外部設計の変更
- 内部構造の変更
- 実装

Step 7-3-1: 外部設計の変更

- ゲーム画面のデザイン + 遊び方



Step 7-3-2: 内部構造の変更

playingタブ

敵機のパラメータ(変数)定義

敵弾のパラメータ(変数)定義

自機のパラメータ(変数)定義

自弾のパラメータ(変数)定義

```
void playing() {
```

ステータス表示処理

敵機の処理(表示・移動)

敵弾の処理(表示・移動・発射)

自機の処理(表示・移動)

自弾の処理(表示・移動・破壊)

キー入力処理

```
}
```

注)敵弾の「発射」

注)自弾の「発射」

敵弾について
追加部分を確認

Step 7-3-3: 実装

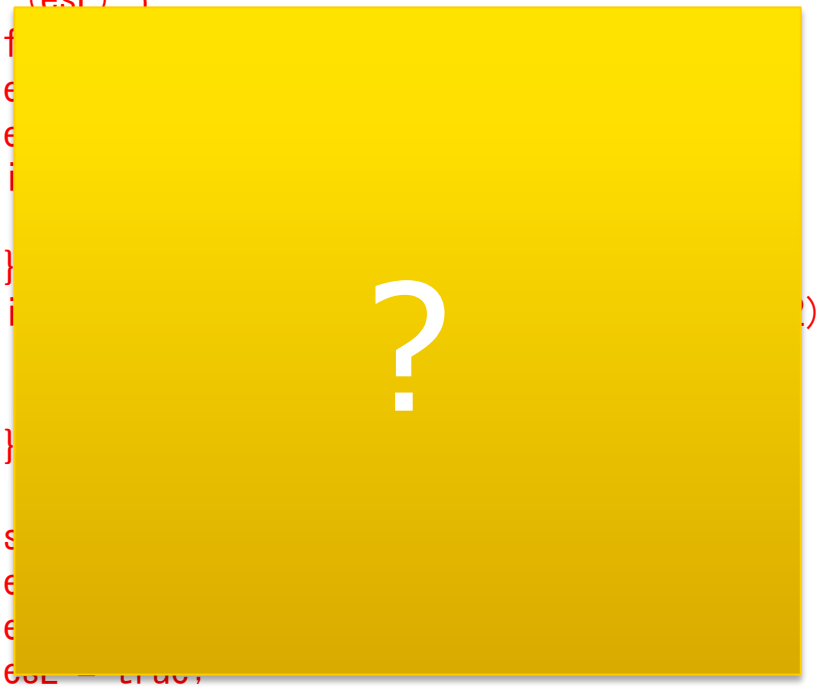
敵弾のパラメータ部分

```
// 敵弾のパラメータ
int esS = 15;    // サイズ
int             座標
int             座標
int             速度
boo             // 発射状態
```

基本的に自弾の
部分をコピーして変更

敵弾処理部分

```
// 敵弾
if (esL) {
  f
  e
  e
  i

  }
  i                                    ) {


  }
}
els
  e
  e
  esL = true;
}
```



Step 7-3-3: 実装

敵弾のパラメータ部分

```
// 敵弾のパラメータ
int esS = 15;   // サイズ
int esX = 200;  // X座標
int esY = 50;   // Y座標
int esV = -3;   // 速度
boolean esL = false; // 発射状態
```

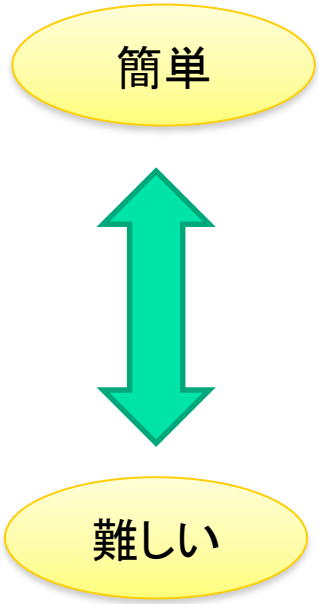
基本的に自弾の
部分をコピーして変更

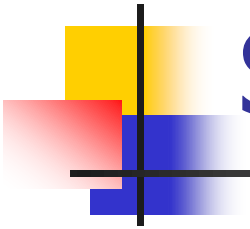
敵弾処理部分

```
// 敵弾
if (esL) {
  fill(255);
  ellipse(esX, esY, esS, esS);
  esY -= esV;
  if (esY > height) {
    esL = false;
  }
  if (dist(pX, pY, esX, esY) < (pS + esS) / 2) {
    background(255, 255, 0);
    scene = 2;
  }
}
else {
  esX = eX;
  esY = eY;
  esL = true;
}
```

Step 7-4: 高機能化

- たとえば …
  - 自機を前後にも動けるように
  - 自機の移動範囲を制限する
  - 複数の敵機が出てくるように
  - 障害物を設置する
  - 弾を同時に何発も撃てるように





Step 7-5: 飾り付ける

- たとえば …
  - 自機、敵機、自弾、敵弾の色を変える
  - 自機や敵機を画像データで表示
  - 弾が当たった時に効果をつける
  - 効果音、BGMをつける
  - 結果画面の表示を派手にする