```
// 星の位置と速度を管理する変数
float x;
float y;
float xspeed;
float yspeed;

// パドルの位置
float paddleY;
float paddleR = 50;

// ポイント
int playerPoint = 0;

float lastReflectionTime = 0;
// 反射から次の反射までの許容時間
float reflectionCooldown = 0.2;
float prevMouseX = 0; // 前回のフレームでの mouseX の位置
float prevMouseY = 0; // 前回のフレームでの mouseY の位置

void setup() {
size(400, 600);
resetStar();
}

void resetStar() {
x = random(0, width); // ランダムな位置から星が出現するようにする
y = random(-50, 0); // 上から出現するように設定
xspeed = random(-5, 5);
yspeed = random(3, 5); // 下に落ちるように設定
}

void draw() {
background(255); // 背景を白色に設定

// 星の位置を更新
x += xspeed;
y += yspeed;

// 壁で跳ね返り
if (x > width - paddleR || x < paddleR) {
xspeed *= -1;
}

float paddleXSpeed = mouseX - prevMouseX; // パドルの x 方向の速度
float paddleYSpeed = mouseY - prevMouseY; // パドルの y 方向の速度
```

```
// パドルで跳ね返り
float distToPlayerPaddle = dist(x, y, mouseX, mouseY);
float currentTime = millis() / 1000.0; // 現在の時間を秒単位で取得

if (distToPlayerPaddle <= paddleR * 2 && currentTime - lastReflectionTime > reflectionCooldown) {
float angle = atan2(y - mouseY, x - mouseX);
float speed = sqrt(xspeed * xspeed + yspeed * yspeed);
xspeed = (speed * 0.8 + abs(paddleXSpeed)) * cos(angle);
yspeed = (speed * 0.8 + abs(paddleYSpeed)) * sin(angle);
lastReflectionTime = currentTime; // 最後の反射の時間を更新
}

// 星が上の画面外に出たらポイントを増加
if (y < -paddleR) {
playerPoint += 1;
resetStar();
}
// 星が下の画面外に出たらポイントを減少
else if (y > height + paddleR) {
playerPoint -= 1;
resetStar();
}

prevMouseX = mouseX; // 現在の mouseX の位置を保存
prevMouseY = mouseY; // 現在の mouseY の位置を保存

// プレイヤーのパドルを表示
fill(0, 255, 0); // 緑色に設定
ellipse(mouseX, mouseY, paddleR * 2, paddleR * 2);

// 星を表示
fill(255, 0, 0); // 赤色に設定
drawStar(x, y, paddleR, paddleR / 2, 5);

// 天井に当たった回数を表示
fill(0); // 黒色に設定
text("POINT: " + playerPoint, 10, 20);
}

// 星を描画する関数
void drawStar(float cx, float cy, float outerRadius, float innerRadius, int numPoints) {
float angle = TWO_PI / numPoints;
float halfAngle = angle / 2.0;
beginShape();
```

```
for (float a = 0; a < TWO_PI; a += angle) {
float sx = cx + cos(a) * outerRadius;
float sy = cy + sin(a) * outerRadius;
vertex(sx, sy);
sx = cx + cos(a + halfAngle) * innerRadius;
sy = cy + sin(a + halfAngle) * innerRadius;
vertex(sx, sy);
}
endShape(CLOSE);
```