Game.pde

```
/*Game Development Class Project by:
 - Rusty Spendlove
 - Malcolm Kyle
 - Kai Li Cantwell
 - Dave Martinez Valencia

 Notes:
 - All graphics are inspired by Ao Oni.
 - Some game mechanics and code logic were implemented with the help of ChatGPT. */
PImage[] marioFrames = new PImage[5];
PImage[] marioFramesBack = new PImage[5];

Room foyer;
Room kitchen;
Room libraryRoom;
//Room rightRoom;

Room currentRoom;

int roomIndex = 0;
Room[] rooms;


Player player;
Key gameKey;
Task task;
Enemy enemy;

PImage keyImg;
PImage enemyImg;
PImage winImg;
PImage loseImg;
PImage pauseImg;
PImage startImg;
PImage backgroundImg;
PImage doorImg;

PImage kitchenImg;
PImage libraryImg;
```

```
PImage foyerImg;


boolean gamePaused = false;
boolean gameWon = false;
boolean gameLost = false;
boolean gamestart = false;


int spawnProtection = 60;


void setup() {
  fullScreen();
  background(0);

  // Load Mario frames
  for (int i = 0; i < 5; i++) {
    marioFrames[i] = loadImage("mario" + (i+1) + ".png");
    marioFramesBack[i] = loadImage("backmario" + (i+1) + ".png");
  }

  // Images
  keyImg = loadImage("key.png");
  enemyImg = loadImage("maltigi better.png");
  winImg = loadImage("win.png");
  loseImg = loadImage("WEGALOSE-1-1.png");
  pauseImg = loadImage("PauseScreen-1-1.png");
  startImg = loadImage("start .png");
  backgroundImg = loadImage("background.png");
  doorImg = loadImage("door.png");

  kitchenImg = loadImage("k2.png");
  libraryImg = loadImage("Library-1.png.png");
  foyerImg = loadImage("Foyer.png");

  player = new Player(this, width/2, height/2, marioFrames, marioFramesBack);


  foyer = new Room(
    this,
    foyerImg, doorImg,
    430, 420, 190, 270, // door x,y,w,h
    new Key(this, 600, 200, keyImg),
```

```
    new Task(this, 50, 50, "Take the first key from NPC"),
    2, 200, // enemy speed & range
    550, 1370, 140, 870, // room boundaries (minX, maxX, minY, maxY)
    enemyImg, 1500, 250, 64, 96   // enemy image and starting x,y,width,height
    );

  kitchen = new Room(
    this, // PApplet
    kitchenImg, doorImg, // background & door images
    480, 440, 190, 260, // door x, y, w, h
    new Key(this, 700, 300, keyImg),
    new Task(this, 50, 50, "Find kitchen key"),
    2.5, 260, // enemy speed & range
    100, width-100, 120, height-120, // room boundaries
    enemyImg, 1500, 250, 64, 96        // enemy image and starting x,y,width,height
    );

  libraryRoom = new Room(
    this, // PApplet
    libraryImg, doorImg, // background & door images
    500, 450, 190, 260, // door x, y, w, h
    new Key(this, 620, 260, keyImg),
    new Task(this, 50, 50, "Solve puzzle"),
    3, 320, // enemy speed & range
    100, width-100, 120, height-120, // room boundaries
    enemyImg, 1500, 250, 64, 96        // enemy image and starting x, y, width, height
    );


  rooms = new Room[] { foyer, kitchen, libraryRoom };
  currentRoom = rooms[0];

  // Reset player to center of current room
  player.x = (currentRoom.minX + currentRoom.maxX) / 2;
  player.y = (currentRoom.minY + currentRoom.maxY) / 2;

  // Reset enemy position inside the current room's wander area
  currentRoom.resetEnemyToWanderZoneCenter();
}

void draw() {
  // Background & room drawing uses currentRoom
  imageMode(CORNER);
```

```
  currentRoom.drawRoom();
//Dave Dave Martinez Valencia, Rusty Spendlove (graphics)
  if (!gamestart) {
    imageMode(CENTER);
    image(startImg, width/2, height/2, width, height);
    imageMode(CORNER); // reset so nothing else breaks
    return;
  }

//Malcom Kyle
  if (gamePaused) {
    imageMode(CENTER);
    image(pauseImg, width/2, height/1.5);
    return;
  }
//Rusty Spendlove, Malcolm Kyle (graphics)
  if (gameWon) {
    imageMode(CENTER);
   // image(winImg, width/2, height/2);
    return;
  }
//Kai Li Cantwell, Malcolm Kyle (graphics)
  if (gameLost) {
    imageMode(CENTER);
    image(loseImg, width/2, height/2);
    return;
  }

 player.update(currentRoom.obstacles, currentRoom.minX, currentRoom.maxX,
currentRoom.minY, currentRoom.maxY);
  player.display();


  currentRoom.roomKey.display();
  currentRoom.roomKey.checkCollision(player);

  if (currentRoom.roomKey.isCollected) currentRoom.roomTask.complete = true;

  currentRoom.roomTask.display();
  currentRoom.roomTask.checkInteraction(player);

//Makes sure enemy's speed and range is set to current room
  currentRoom.roomEnemy.speed = currentRoom.enemySpeed;
  currentRoom.roomEnemy.range = currentRoom.enemyRange;
```

```
 )
  currentRoom.roomEnemy.update(player, currentRoom.obstacles,
currentRoom.getWanderZone());
  currentRoom.roomEnemy.display();

  // If task & key complete and player at door, move to next room
  if (currentRoom.roomTask.complete && currentRoom.roomKey.isCollected) {
    if (currentRoom.playerAtDoor()) {
      roomIndex++;

      if (roomIndex >= rooms.length) {
        gameWon = true;
        return;
      }

      currentRoom = rooms[roomIndex];

      // Reset player to center of new room
      player.x = (currentRoom.minX + currentRoom.maxX) / 2;
      player.y = (currentRoom.minY + currentRoom.maxY) / 2;

      // reset enemy position to the center of the new room's wander zone
      currentRoom.resetEnemyToWanderZoneCenter();

      // Reset spawn protection so player doesn’t instantly die
      spawnProtection = 60;
    }
  }

  // Spawn protection prevents instant kill after starting in a new room
  if (spawnProtection > 0) {
    spawnProtection--;
  } else {
    if (currentRoom.roomEnemy.checkCollision(player)) {
      gameLost = true;
    }
  }
}

void keyPressed() {
  if (key == 'p' || key == 'P') gamePaused = !gamePaused;

  if (!gamePaused && !gameWon && !gameLost) {
```

```
    if (key == 'a' || key == 'A') player.movingLeft = true;
    if (key == 'd' || key == 'D') player.movingRight = true;
    if (key == 'w' || key == 'W') player.movingUp = true;
    if (key == 's' || key == 'S') player.movingDown = true;
  }
}

void keyReleased() {
  if (!gamePaused && !gameWon && !gameLost) {
    if (key == 'a' || key == 'A') player.movingLeft = false;
    if (key == 'd' || key == 'D') player.movingRight = false;
    if (key == 'w' || key == 'W') player.movingUp = false;
    if (key == 's' || key == 'S') player.movingDown = false;
  }
}

void mousePressed() {
  if (!gamestart) {
    gamestart = true;
    spawnProtection = 60; // RESET protection when start is pressed
  }
}
```

Enemy Class

```
// Dave Martinez Valencia, Kai Li Cantwell
//(Some game mechanics and code logic were implemented with the help of ChatGPT)
class Enemy {
  PApplet parent;
  PImage img;
  float x, y, w, h;
  float speed;
  float range;
  float dirX, dirY;
  float chaseMultiplier = 1.2f; // tuned for smoother chase
  float wanderMultiplier = 0.6f; // slower wandering

  float hitboxXOffset = 7;
  float hitboxYOffset = 5;
  float hitboxWidth = 40;
  float hitboxHeight = 60;
```

```
int wanderTimer = 0;
int wanderDelay = 60;

// Current movement area limits
float leftLimit, rightLimit, topLimit, bottomLimit;


Enemy(PApplet p, float startX, float startY, float w_, float h_, float spd, float rng, PImage i) {
  parent = p;
  x = startX;
  y = startY;
  w = w_;
  h = h_;
  speed = spd;
  range = rng;
  img = i;
  dirX = parent.random(-1, 1);
  dirY = parent.random(-1, 1);
  // defaults - will be overwritten by Room constructor
  leftLimit = 0;
  rightLimit = parent.width;
  topLimit = 0;
  bottomLimit = parent.height;
}


void update(Player p, ArrayList<Obstacle> obstacles, float[] wanderZone) {
  float moveX = 0, moveY = 0;

  float distance = parent.dist(
    x + w/2, y + h/2,
    p.x + p.width/2,
    p.y + p.height/2
    );

  // Enemy Chase function
  if (distance < range) {
    float angle = parent.atan2(
      p.y + p.height/2 - (y + h/2),
      p.x + p.width/2 - (x + w/2)
      );
    moveX = parent.cos(angle) * speed * chaseMultiplier;
    moveY = parent.sin(angle) * speed * chaseMultiplier;
```

```
}
//Enemy wander function
else {
  wanderTimer++;
  if (wanderTimer >= wanderDelay) {
    dirX = parent.random(-1, 1);
    dirY = parent.random(-1, 1);
    wanderTimer = 0;
  }
  moveX = dirX * speed * wanderMultiplier;
  moveY = dirY * speed * wanderMultiplier;
}

float oldX = x;
float oldY = y;

// Checks for obstacle collision when along moving X
x += moveX;
for (Obstacle o : obstacles) {
  if (collidesWith(o.x, o.y, o.w, o.h)) {
    x = oldX;
    dirX *= -1;
    break;
  }
}

// Checks for obstacle collision when along moving Y
y += moveY;
for (Obstacle o : obstacles) {
  if (collidesWith(o.x, o.y, o.w, o.h)) {
    y = oldY;
    dirY *= -1;
    break;
  }
}

// Enforce wandering zone
if (wanderZone != null && wanderZone.length == 4) {
  float wzMinX = wanderZone[0];
  float wzMaxX = wanderZone[1];
  float wzMinY = wanderZone[2];
  float wzMaxY = wanderZone[3];

  if (x < wzMinX) {
```

```
      x = wzMinX;
      dirX *= -1;
    }
    if (x + w > wzMaxX) {
      x = wzMaxX - w;
      dirX *= -1;
    }
    if (y < wzMinY) {
      y = wzMinY;
      dirY *= -1;
    }
    if (y + h > wzMaxY) {
      y = wzMaxY - h;
      dirY *= -1;
    }
  } else {
    // fallback to full room limits
    if (x < leftLimit) {
      x = leftLimit;
      dirX *= -1;
    }
    if (x + w > rightLimit) {
      x = rightLimit - w;
      dirX *= -1;
    }
    if (y < topLimit) {
      y = topLimit;
      dirY *= -1;
    }
    if (y + h > bottomLimit) {
      y = bottomLimit - h;
      dirY *= -1;
    }
  }
}

void display() {
  parent.image(img, x, y, w, h);
  // debug hitbox (visible for development)
  parent.noFill();
  parent.stroke(255, 0, 0);
  parent.rect(x + hitboxXOffset, y + hitboxYOffset, hitboxWidth, hitboxHeight);
}
```

```
  boolean collidesWith(float ox, float oy, float ow, float oh) {
    float hx = x + hitboxXOffset;
    float hy = y + hitboxYOffset;
    return !(hx + hitboxWidth <= ox || hx >= ox + ow || hy + hitboxHeight <= oy || hy >= oy + oh);
  }

  boolean checkCollision(Player p) {
    // Use player's hitbox vs enemy's hitbox for accurate collision
    float ex = x + hitboxXOffset;
    float ey = y + hitboxYOffset;
    return !(p.x + p.hitboxWidth < ex || p.x + p.hitboxXOffset > ex + hitboxWidth || p.y +
p.hitboxHeight < ey || p.y + p.hitboxYOffset > ey + hitboxHeight);
  }
}
```

Key Class

```
// Malcom Kyle
class Key {
  PApplet parent;
  float x, y;
  float size = 40;
  boolean isCollected = false;
  PImage img;

  Key(PApplet p, float startX, float startY, PImage keyImage) {
    parent = p;
    x = startX;
    y = startY;
    img = keyImage;
  }

  void display() {
    if (!isCollected) {
      parent.imageMode(CENTER);
      parent.image(img, x, y, size, size);
    }
  }

  void checkCollision(Player p) {
    if (!isCollected) {
```

```
      float px = p.x + p.hitboxXOffset;
      float py = p.y + p.hitboxYOffset;
      if (!(px + p.hitboxWidth <= x - size/2 || px >= x + size/2 || py + p.hitboxHeight <= y - size/2 ||
py >= y + size/2)) {
        isCollected = true;
      }
    }
  }
}
```

Obstacle Class

```
// Dave Martinez Valencia
class Obstacle {
  float x, y, w, h;
  Obstacle(float x, float y, float w, float h) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
  }
  void display() {
    noFill();
    noStroke();
    rect(x, y, w, h);
  }
}
```

Player Class

```
// Dave Martinez Valencia
//(Some game mechanics and code logic were implemented with the help of ChatGPT)
class Player {
  PApplet parent;

  PImage[] marioFrames;
  PImage[] backMarioFrames;
  PImage[] currentFrames;

  int frameCount;
  int currentFrame = 0;
```

```
int frameDelay = 2;
int frameTimer = 0;

float x, y;
float vx = 0, vy = 0;
float speed = 10;

float width = 64, height = 96;

boolean movingLeft = false;
boolean movingRight = false;
boolean movingUp = false;
boolean movingDown = false;

// Hitbox (offsets from player's x,y)
float hitboxXOffset = 10;
float hitboxYOffset = -5;
float hitboxWidth = 40;
float hitboxHeight = 70;

Player(PApplet p, float startX, float startY, PImage[] front, PImage[] back) {
  parent = p;
  x = startX;
  y = startY;
  marioFrames = front;
  backMarioFrames = back;
  currentFrames = marioFrames;
  frameCount = marioFrames.length;
}

// Pass in obstacles and room limits
void update(ArrayList<Obstacle> obstacles, float minX, float maxX, float minY, float maxY) {
  float oldX = x;
  float oldY = y;
  vx = vy = 0;

  if (movingLeft)  vx = -speed;
  if (movingRight) vx = speed;
  if (movingUp)    vy = -speed;
  if (movingDown)  vy = speed;

  x += vx;
  // Check horizontal collisions with obstacles
```

```
  for (Obstacle o : obstacles) {
    if (collidesWith(o.x, o.y, o.w, o.h)) {
      x = oldX;
      break;
    }
  }

  y += vy;
  // Check vertical collisions with obstacles
  for (Obstacle o : obstacles) {
    if (collidesWith(o.x, o.y, o.w, o.h)) {
      y = oldY;
      break;
    }
  }

  // Enforce room boundaries (invisible walls)
  x = parent.constrain(x, minX, maxX - width);
  y = parent.constrain(y, minY, maxY - height);

  // Animation switching
  currentFrames = movingRight ? backMarioFrames : marioFrames;

  // Animation logic
  if (vx != 0 || vy != 0) {
    frameTimer++;
    if (frameTimer >= frameDelay) {
      currentFrame = (currentFrame + 1) % frameCount;
      frameTimer = 0;
    }
  } else {
    currentFrame = 0;
  }
}

void display() {
  parent.image(currentFrames[currentFrame], x, y, width, height);
  // debug hitbox
  parent.noFill();
  parent.stroke(255, 0, 0);
  parent.rect(x + hitboxXOffset, y + hitboxYOffset, hitboxWidth, hitboxHeight);
}

boolean collidesWith(float ox, float oy, float ow, float oh) {
```

```
    float hx = x + hitboxXOffset;
    float hy = y + hitboxYOffset;
    return !(hx + hitboxWidth <= ox || hx >= ox + ow || hy + hitboxHeight <= oy || hy >= oy + oh);
  }
}
```

Room Class

```
//Kai Li Cantwell
//(Some game mechanics and code logic were implemented with the help of ChatGPT)
class Room {
  PApplet parent;

  PImage bg;
  PImage doorImg;
  float doorX, doorY, doorW, doorH;

  float minX, maxX, minY, maxY;

  Key roomKey;
  Task roomTask;
  Enemy roomEnemy;

  float enemySpeed;
  float enemyRange;

  ArrayList<Obstacle> obstacles;

  // Determines how much of the room the Enemy is allowed to wander in
  final float WANDER_ZONE_PADDING = 0.25f;

  Room(PApplet p, PImage bg, PImage doorImg,
    float doorX, float doorY, float doorW, float doorH,
    Key key, Task task,
    float enemySpeed, float enemyRange,
    float minX, float maxX, float minY, float maxY,
    PImage enemyImg, float enemyStartX, float enemyStartY, float enemyW, float enemyH) {
    parent = p;

    this.bg = bg;
    this.doorImg = doorImg;
```

```
    this.doorX = doorX;
    this.doorY = doorY;
    this.doorW = doorW;
    this.doorH = doorH;

    this.roomKey = key;
    this.roomTask = task;

    this.enemySpeed = enemySpeed;
    this.enemyRange = enemyRange;

    this.minX = minX;
    this.maxX = maxX;
    this.minY = minY;
    this.maxY = maxY;

    this.obstacles = new ArrayList<Obstacle>();

    // Create a room-specific enemy
    this.roomEnemy = new Enemy(parent, enemyStartX, enemyStartY, enemyW, enemyH,
enemySpeed, enemyRange, enemyImg);

    // Set enemy movement boundaries to room limits
    this.roomEnemy.leftLimit = this.minX;
    this.roomEnemy.rightLimit = this.maxX;
    this.roomEnemy.topLimit = this.minY;
    this.roomEnemy.bottomLimit = this.maxY;
  }

  void drawRoom() {
    imageMode(CORNER);
    parent.image(bg, 0, 0, parent.width, parent.height);
    parent.image(doorImg, doorX, doorY, doorW, doorH);

    // Draw obstacles
    for (Obstacle o : obstacles) o.display();


  }

  boolean playerAtDoor() {
    return player.x + player.width > doorX &&
      player.x < doorX + doorW &&
      player.y + player.height > doorY &&
```

```
      player.y < doorY + doorH;
  }

  // Creates a smaller wander zone for the enemy so that it doesn't
// constantly hit walls and obstacles or get stuck in corners
  float[] getWanderZone() {
    float widthRoom = maxX - minX;
    float heightRoom = maxY - minY;
    float padX = widthRoom * WANDER_ZONE_PADDING;
    float padY = heightRoom * WANDER_ZONE_PADDING;
    float wzMinX = minX + padX;
    float wzMaxX = maxX - padX;
    float wzMinY = minY + padY;
    float wzMaxY = maxY - padY;
    return new float[] { wzMinX, wzMaxX, wzMinY, wzMaxY };
  }

  // Place enemy at center of the wander zone
  void resetEnemyToWanderZoneCenter() {
    float[] wz = getWanderZone();
    roomEnemy.x = (wz[0] + wz[1]) / 2.0;
    roomEnemy.y = (wz[2] + wz[3]) / 2.0;
    // reset wander direction randomly
    roomEnemy.dirX = parent.random(-1, 1);
    roomEnemy.dirY = parent.random(-1, 1);
  }
}
```

Task Class

```
//Rusty Spendlove
class Task {
  PApplet parent;
  float x, y;
  String description;
  boolean complete = false;
  float size = 40;

  Task(PApplet p, float startX, float startY, String desc) {
    parent = p;
    x = startX;
    y = startY;
```

```
    description = desc;
  }

  void display() {
    parent.textAlign(LEFT, TOP);
    parent.textSize(28);
    if (complete) {
      parent.fill(100, 255, 100); // green when complete
    } else {
      parent.fill(255); // white otherwise
    }
    parent.text(description, 50, 50);
  }

  void checkInteraction(Player p) {
    // simple proximity using player's center and task size
    float distance = parent.dist(x, y, p.x + p.width/2, p.y + p.height/2);
    if (distance < 50) complete = true;
  }
}
```