PACMAN

Was wir bisher haben:

Main:

```
float rectSize = 40;
Pacman pac;

void settings() {
    size(400, 400);
}


void setup() {
    background(0);

    // erstelle pacman
    pac = new Pacman(width/2, height/2, rectSize);
}


void draw() {
    // zeichne hintergrund
    background(0);
        // zeichne pacman
        pac.paint();
}


void keyPressed() {
    if (keyCode == UP) {
            pac.moveUp();
    } else if (keyCode == DOWN) {
            pac.moveDown();
    } else if (keyCode == LEFT) {
            pac.moveLeft();
    } else if (keyCode == RIGHT) {
            pac.moveRight();
    }
}



class Pacman{
    color yellow = color(255, 255, 0);
    int speed = 8;
    float diam;
    float x;
    float y;

    //Konstruktor
    Pacman(float x, float y, float size){
        this.x = x;
        this.y = y;
        this.diam = size;
    }

    void paint(){
        noStroke();
        fill(yellow);
        ellipse(x, y, diam, diam);
    }
```

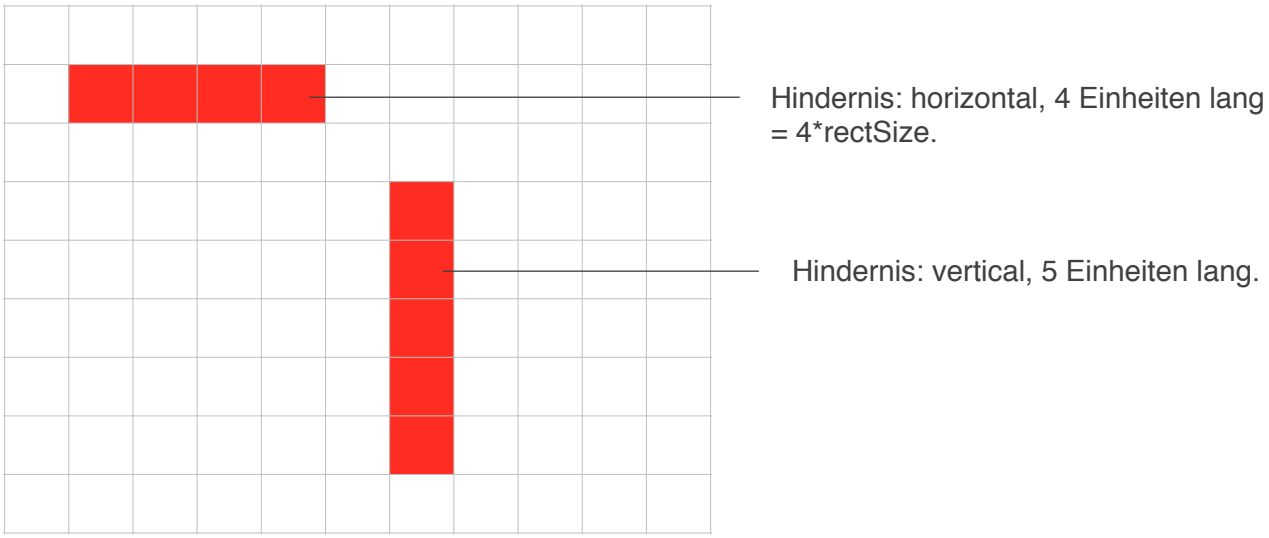
```
    void moveLeft(){
        x = x + -1*speed;
    }

    void moveRight(){
        x = x + speed;
    }

    void moveUp(){
        y = y + -1*speed;
    }

    void moveDown(){
        y = y + speed;
    }

    float getX(){
        return x;
    }

    float getY(){
        return y;
    }

    float getSize(){
        return diam;
    }
}
```

Wie wir unser Spielfeld aufbauen wollen:

Um das Labyrinth an Hindernissen und die Nuggets die es einzusammeln gilt zu zeichnen, unterteilen wir unser Spielfeld im Kopf in ein Grid. Jedes Gridelement hat dabei die Grösse rectSize x rectSize, welches unsere Grundeinheit darstellen wird:

Spielfeld:

Nun können wir Hindernisse erstellen, welche eine Breite von rectSize haben und eine Länge von x*rectSize. Sie können vertical oder horizontal angeordnet sein.



AUFGABE 1 - Hindernisse

Wir wollen Hindernisse / Labyrinthwände erstellen durch die Pacman (und später auch die Gegner) nicht durchfahren können.
Ergänze dazu die folgende Klasse für die Hindernisse:

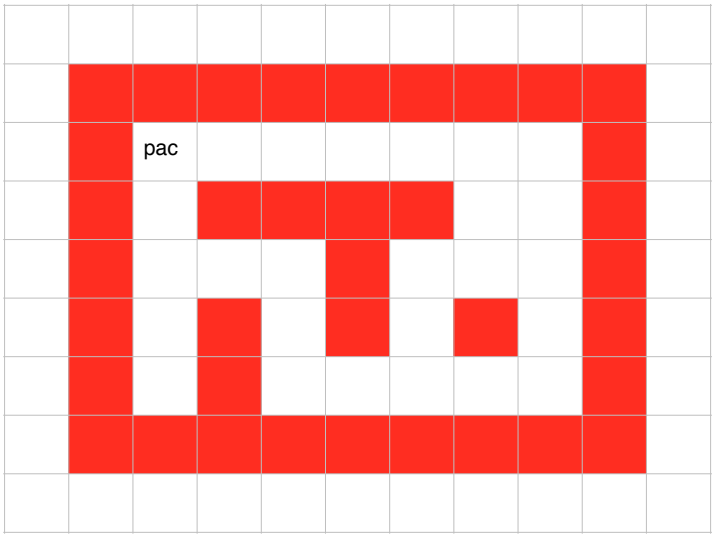
```
class Obstacle{
    // Klassenattribute für Position, Länge und Breite.
    // Und isVertical zum definieren ob das Rechteck liegend oder stehend
gezeichnet wird.


    // Konstruktor
    Obstacle(float x, float y, boolean isVertical, float obLength, float
rectSize){
        // Initialisieren die Position und ob das Hindernis vertical oder
horinzontal liegt.
        // Initialisiere die Hindernis Breite und Länge. Diese sind abhängig
davon wie das Hindernis liegt.
        // Die längere Seite soll in Einheiten, nicht der effektiven Länge,
übergeben werden und muss hier nun wieder in
        // eine Länge umgerechnet werden also: obLength * rectSize
    }

    // Methode zum zeichnen des Rechtecks
    void paint(){
    }

    // Methode die true zurück gibt, wenn die x UND y Parameter das Rechteck
berühren
    // oder innerhalb des Rechtecks liegen.
    boolean isTouched(float x, float y){
    }

}
```

Erstelle nun einen Array von Hindernissen in Main, lasse sie zeichnen und teste ob dein Pacman nicht drüber fahren kann.

Sobald das klappt, kannst du eine Klasse Playfield erstellen, welche ab jetzt die Hindernisse verwalten soll. Ergänze dazu diese Klasse:

```
class PlayField {
    // Attribut für die Hindernisse (Array von Hindernissen)
    // und die rectSize

    PlayField(float rectSize) {
        this.rectSize = rectSize;
        initObstacles();
    }


    void initObstacles() {
        // erstelle hier deine Hindernisse
    }


    void paint(){
        // zeichne die Hindernisse
    }


    boolean isObstacleTouched(float x, float y){
        // prüfe ob eines der Hindernisse berührt wurde
    }

}
```

Zusatz: Erstelle deine Hindernisse so, dass dein Spielfeld so aussieht:
Und setzte deinen Pacman in die linke obere Ecke des Spielfelds.



AUFGABE 2 - Score

Nun wollen wir die Elemente erstellen, die Pacman einsammeln kann und somit seinen score erhöhen. Ich nenne diese Elemente Nuggets. Ein Nugget soll als kleiner blauer Kreis gezeichnet werden und die Methode `boolean getsEaten(float x, float y)` besitzen. Diese funktioniert ähnlich wie die `isTouched()` Methode, sobald Pacman ein Nugget berührt wird es „gegessen“. Wenn ein Nugget gegessen wurde, soll es nicht mehr gezeichnet also nicht mehr sichtbar sein.
Ergänze dazu die folgende Klasse:

```
class Nugget{
```

```
    // Attribute für Position und Grösse (setzte Grösse = 5)
    // Attribut für visible Zustand. Also Attribut das speichern kann ob ein
Nugget visible ist oder nicht

    Nugget(float x, float y){
        this.x = x;
        this.y = y;
    }


    void paint(){
        // zeichne einen blauen Kreis, falls das Nugget visible ist
    }


    boolean getsEaten(float x, float y){
        // teste ob das Nugget gegessen wird / ob der Pacman das Nugget berührt
        // vergesse nicht, dass es nicht mehr gegessen werden kann, wenn es
nicht visible ist
    }


    void setVisible(boolean visible){
        // eine Methode um das visible Attribut zu setzen.
    }

}
```

Erstelle eine globale `score` Variable im Main script, welche erhöht wird, wenn ein Nugget gefressen wird. Mach deinen Score sichtbar indem du ihn mit `text()` immer anzeigst. Erstelle nun ein Nugget Objekt und teste ob alles funktioniert.

Füge der Playfield Klasse nun auch einen Array von Nuggets hinzu und ergänze die Klasse mit einer Methode, die prüfen kann ob ein Nugget gegessen wird und dieses dann invisible setzt:

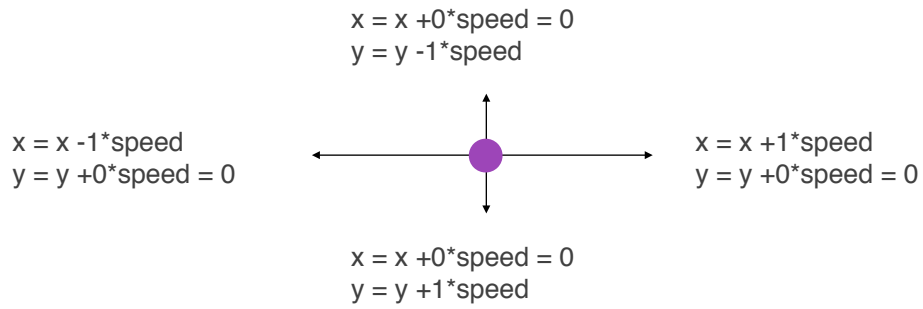
```
class PlayField {
    float rectSize;
    int nrOfElementsWidht = 9;
    int nrOfElementsHeight = 7;
    float startFieldX;
    float startFieldY;

    Obstacle[] obstacles;
    Nugget[] nuggets = new Nugget[nrOfElementsWidht*nrOfElementsHeight];

    PlayField(float rectSize) {
        this.rectSize = rectSize;
        this.startFieldX = rectSize;
        this.startFieldY = rectSize;

        initObstacles();
        initNuggets();
    }


    void initObstacles() {
        // haben wir schon
    }

    // überneheme diese methode um die Nuggets zu zeichnen
    // in jedem Gridelement wird ein Nugget in die Mitte Plaziert
    void initNuggets(){
        int nuggetCount = 0;
        for ( int x = 0; x < nrOfElementsWidht; x++ ) {
```

```
            for ( int y = 0; y < nrOfElementsHeight; y++ ) {
                nuggets[nuggetCount] = new
Nugget(startFieldX+x*rectSize+rectSize/2, startFieldY+y*rectSize+rectSize/2);
                nuggetCount++;
            }
        }
    }


    void paint(){
        // zeichne zuerst die Nuggets und dann die Hindernisse
    }


    boolean isOneNuggetEaten(float x, float y){
        // prüfe ob eines der Nuggets gegessen wird, falls ja, setzte dieses
invisible
    }

}
```

Ändere das Main script entsprechend.

AUFGABE 3 - Gegner

Nun wollen wir Gegner mit einer kleinen AI erstellen. Dazu brauchen wir eine Enemy Klasse, mit einer `move()` Methode. Diese wird für jeden `draw()` Durchlauf vor `Enemy.paint()` aufgerufen. Die move() Methode lässt, den Gegner sich vertikal oder horizontal bewegen, solange er nicht auf ein Hindernis stösst. Das heisst wir ändern die x und y Koordinaten bei jedem move Aufruf. Dazu brauchen wir ein deltaX und deltaY, welche speichern in welche Richtung sich der Gegner bewegt. Sie können die Werte -1, 0, 1 annehmen. Ist deltaX = -1, dann bewegt sich der Gegner nach links, da sich x wie folgt verändert: x = x+deltaX*speed, also x = x-1*speed. Wenn deltaX nicht gleich 0 ist, muss deltaY gleich 0 sein, damit eine vertikale Bewegung gewährleistet wird und umgekehrt.




```
class Enemy{
    float x;
    float y;
    int deltaX;t
    int deltaY;
    float speed = 1;
    float enLength;
    float enWidth;
    PlayField obstaclesField;

    Enemy(float x, float y, float rectSize, PlayField field){
        this.x = x;
        this.y = y;
```

```
        this.obstaclesField = field;
        deltaX = 1;
        deltaY = 0;
        this.enLength = rectSize;
        this.enWidth = rectSize;
    }

    // Enemy AI
    void move(){
        // solange(while) ein Hindernis des Playfield berührt wird, soll die
Richtung gewechselt werden -> changeDirection()
        // wenn nichts mehr berührt wird, soll x um deltaX*speed und y um
deltaY*Speed erhöht werden:
      x += deltaX*speed;
      y += deltaY*speed;
    }


    void changeDirection(){
        // weise deltaX einen zufälligen Wert von -1, 0 oder 1 zu
        // falls deltaX == 0 ist, dann weise deltaY zufällig -1 oder 1 zu.
        // falls deltaX != 0 ist, soll deltaY = 0 sein.
    }


    void paint(){
        fill(200, 0, 255);
        ellipse(x,y, enWidth, enLength);
    }


    boolean collides(float x, float y){
        // prüfe ob der Enemy berührt wird
    }
}
```

-> Input ArrayList

Erstelle im Main script eine ArrayList für Enemy.
Erstelle im Main script eine globale Variable `timeCount` welche bei jedem draw() Aufruf erhöht wird. Wenn der timeCount == 200 ist, soll der ArrayList ein neues Enemy Objekt hinzugefügt und der timeCount wieder auf 0 gesetzt werden. Füge nur solange enemies hinzu bis du 6 hast. Zeichne und bewege jeden enemy für jeden draw() Durchlauf und prüfe auch jedesmal ob einer mit Pacman kollidiert. Gib auf der Konsole: „du bist tod“ aus, wenn er kollidiert.

AUFGABE 4 - Spiel abrunden: Game start, Game over

Füge im Main script eine gloable Variable hinzu welche prüft ob Pacman sich schon bewegt hat und eine die prüfen kann ob das Spiel fertig ist, also ein Gegner berührt wurde.
Nutze die erste Variable dazu, dass du erst Gegner generierst, sobald sich Pacman zu bewegen beginnt.
Und nutze die zweite dazu, dass Spiel abzubrechen, wenn ein Gegner berührt wurde. Zeichne dazu nichts mehr vom Spiel sondern zeige nur einen Text „Game Over“ und vielleicht noch welchen Score erreicht wurde.