



КУРСОВ ПРОЕКТ

Игра на понг с Ардуино

Факултет по Математика и Информатика

Студент: Хари Христов Христов



Съдържание:

1. Съдържание
2. Увод
 - 2.1. Значение на играта
 - 2.2. Описание на проекта
3. Проектиране
 - 3.1. Поток на данни
 - 3.2. Алгоритъм
 - 3.3. Блок – схема
4. Прототип
 - 4.1. Бюджет и компоненти
 - 4.2. Разположение и свързване на компонентите
 - 4.3. Демо (снимка)
5. Възможни подобрения
6. Програмен код (Source code)



2. Увод

Целта на всяка игра фундаментално е да бъде средство за разпускане и забавление. Разработката им започва много назад в миналото като в самото начало игрите са били много по примитивни и прости от тези, с които разполагаме и на които сме свикнали в днешно време. Играта Pong е създадена в далечната 1972, като за времето си тя е революционна. Простият ѝ дизайн я прави широкодостъпна и много популярна.

2.1. Значение на играта

Pong играе ключова роля във въвеждането на цифровото забавление пред публика. Той показва потенциала на интерактивните електронни медии и залага основите за еволюцията на видеоигрите в развлекателния сектор. Развитието на играта включва иновативното използване на новонавлезли технологии в областта на графиката и симулацията. Успехът ѝ стимулира напредък в областта на компютърните и графичните технологии и допринася за настъплението на аркадната култура, където хората се събират в аркадни зали, за да играят видеоигри. Този



социален аспект на игрите слага основите за мултиплейър гейминга, който е характерен за съвременните видеоигри.

2.2. Описание на проекта

Проектът представлява една много примитивна, но все пак напълно интерактивна и функционална версия на класическата игра Pong. Той е в компактна форма, тъй като целта му е да може да бъде лесно преносим и да може да се ползва по всяко време и на всяко място. Прототипът е един малък бредборд, на който са поставени и свързани платката с логиката, дисплейят, на който се визуализира играта и бутоните, с които двамата играчи контролират своите „ракети”.

3. Проектиране

3.1. Поток на данни

Данните ще идват от потребителя чрез натискане на някои от 4те бутона – по 2 бутона, отговарящи за една ракета, като единият я движи надолу по екрана, а другият нагоре (по една и съща линия). При всяко натискане на бутон се подава сигнал към платката, който при всяко въртене на loop



функцията проверява дали е получил съответния сигнал и променя положението на съответната ракета (или не прави нищо при липсата на такъв сигнал).

3.2. Алгоритъм

Първоначално топчето започва да се движи в една посока диагонално като началната му точка се намира около средата на дисплея. Алгоритъмът проверява първо дали има подаден сигнал от някой/някои от четирите бутона. При наличие на такъв сигнал се изпълнява следната последователност от операции:

1. цветът на ракетата се променя от бял на черен, с което тя видимо изчезва от екрана
2. координатите на началната точка на ракетите се ъпдейтва спрямо натиснатия бутон
3. рисува се върху екрана същата ракета, но с вече променените начални координати и отново с бял цвят, с което се симулира движението ѝ

След това по същият начин променя мястото на топчето върху екрана(старите координати се оцветяват в черно, а новите се изобразяват с бял цвят). Последната част от алгоритъма е промяната на координатите на топчето, където



също се проверява дали играта е завършила т.е. топчето е ударило някоя от малките страни на дисплея. Тази част от алгоритъма ще е разделена в две части с цел да се бъде представена по-пригледно – първо ще е проверката дали играта е приключила и след това ще е логиката по промяната на координатите на топчето.

Проверка за край на играта

Когато топчето удари някоя от малките стени на екрана играта се счита за приключила, което означава, че екранът замръзва за половин секунда, след което се появява завършващият екран, на който е написано, че играта е свършила и кога ще започне следващата. След това позициите на ракетите и топчето се рестартират към първоначалните си и началният екран се визуализира отново.

Логика за движение на топчето

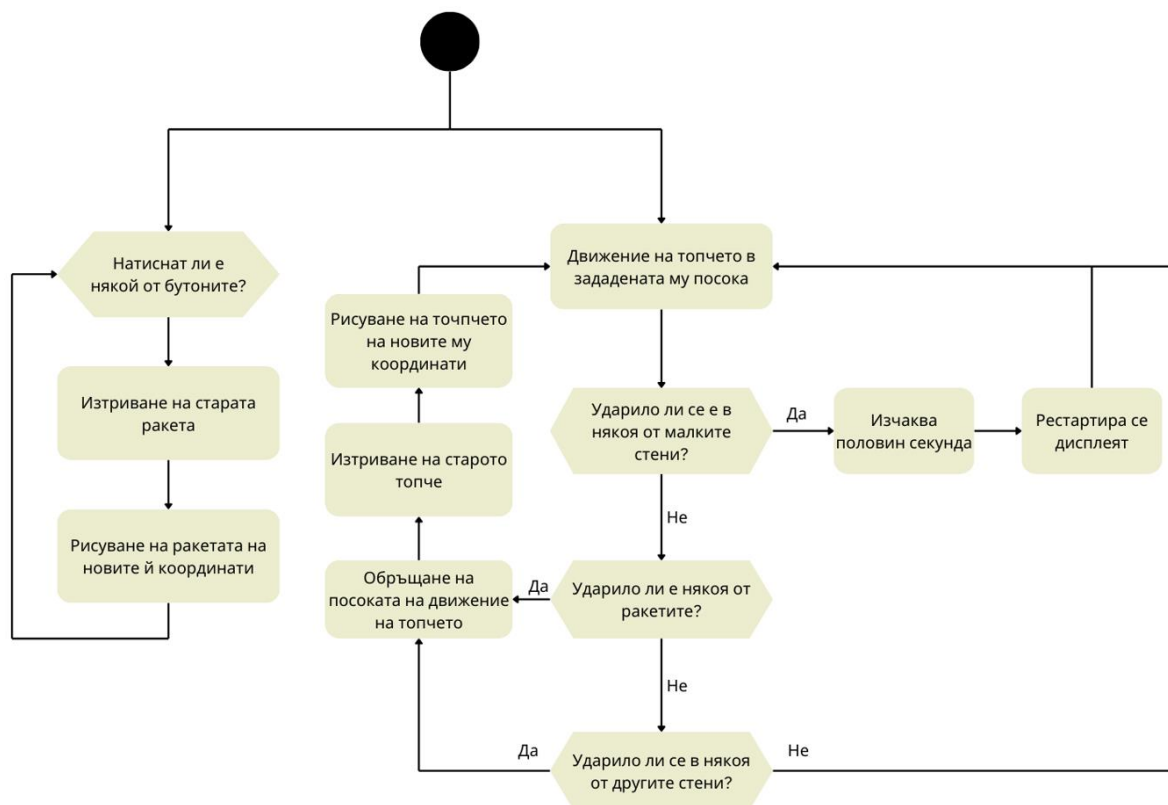
Променяме стойността на старите координати на топчето да бъдат равни на сегашните, след което към сегашните координати добавяме скоростта, с която се движи топчето.

След това валидираме получените координати (проверяваме дали единият/двата координата не се намират извън

очертанията на екрана след местенето и ако са – ги задаваме да бъдат точно на границата и променяме посоката, в която се движи топчето).

3.3. Блок-схема

Следната фигура представя алгоритъма схематично с цел да покаже нагледно целия процес на работа на проекта.

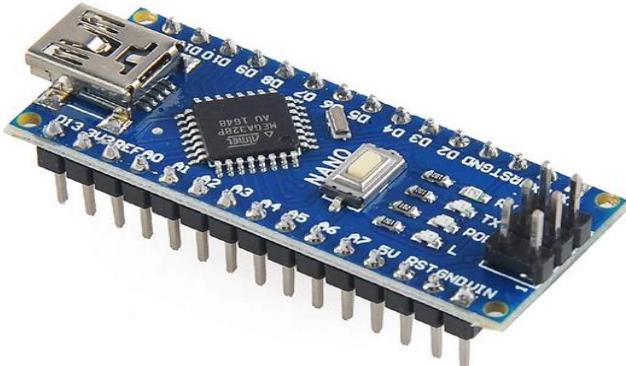
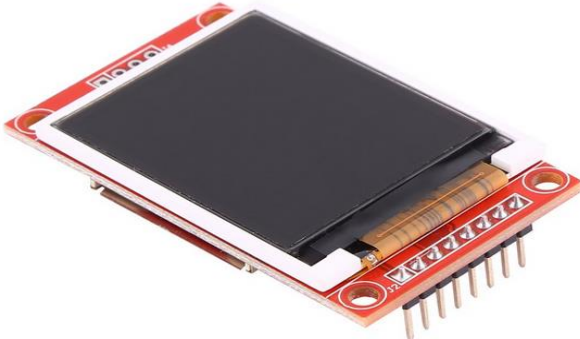



Фиг 3.3

4.Прототип

4.1. Бюджет и компоненти

Цялостният бюджет излиза около 25лв., в зависимост от наличието и сегашната цена на компонентите.

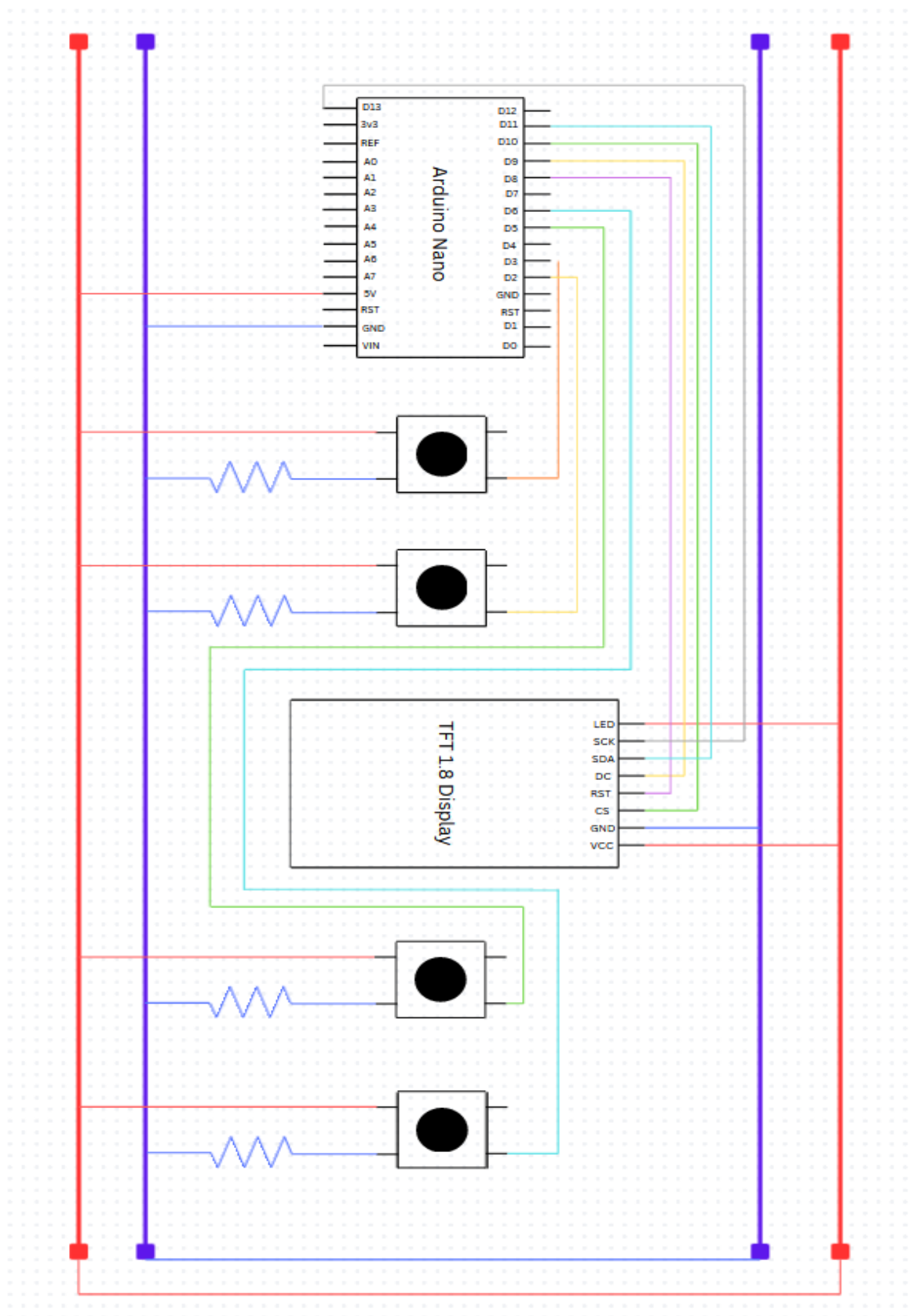
Компонент	Снимка	Цена
Ардуино Нано платка		10лв.
Дисплей TFT 1.8		17лв.
Кабели (стандартни)		5лв.



4 резистора, 220 Ω		0.20лв.
4 бутона		1лв.

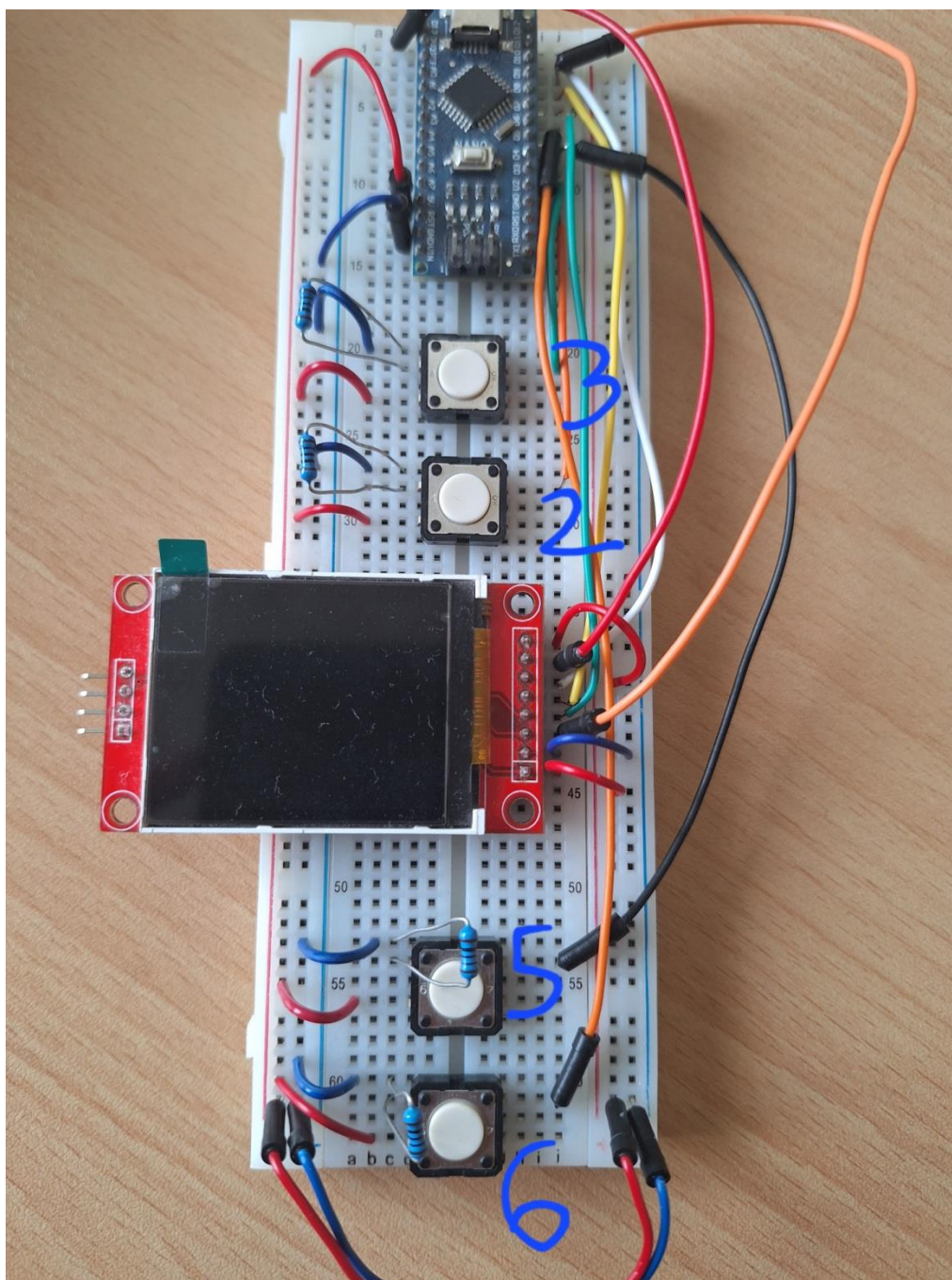
Фиг. 4.1

4.2. Разположение и свързване на компонентите



Фиг. 4.2

4.3. Демо (снимка)



Фиг. 4.3



5. Възможни подобрения

С цел играта да върви по-гладко и да бъде по-приятна за всеки, който я играе смятам, че следните подобрения ще бъдат от полза:

- Добавяне на начален екран, който да приветства играча и да чака той да натисне някой от бутоните, за да започне играта, вместо играта автоматично да започва.
- Имплементиране на таблица с най-високите резултати с цел да се провокира състезателният дух на играчите.
- Добавяне на горен капак към прототипа, което да завърши вида му на портативна конзола и да го направи по-удобен за пренасяне.
- Имплементиране на опцията за Singleplayer, където играчът ще играе срещу бот, което ще премахне необходимостта от двама играча за нормалното протичане на играта.
- Добавяне на батерия, която да премахне нуждата от кабел, който да осигурява захранване на платката.

6. Сурс код (Source code)



```
#include <TFT.h>
#include <SPI.h>

#define cs  10
#define dc  9
#define rst  8
#define leftDownBtn 2
#define leftUpBtn 3
#define rightDownBtn 5
#define rightUpBtn 6

// screen
TFT screen = TFT(cs, dc, rst);
int width, height;

// ball
int defaultSpeed = 4;
int oldBallX, oldBallY;
int ballX, ballY;
int ballSize = 4;
int ballXDirection = defaultSpeed;
int ballYDirection = defaultSpeed;

// paddles
int paddleHeight = 40;
int paddleWidth = 4;
int paddleJump = 5;
int leftY, rightY;
int oldLeftY, oldRightY;
int leftPaddleX;
```



```
int rightPaddleX;

// util
bool isChanged = false;

// setup
void setup()
{
    screen.begin();
    screen.background(0, 0, 0);
    width = screen.width();
    height = screen.height();

    leftY = rightY = oldRightY = oldLeftY = height / 2 - 20;
    leftPaddleX = 0;
    rightPaddleX = width - 4;

    ballX = oldBallX = width / 2 - 12; // to be slightly off-center
    ballY = oldBallY = height / 2 - 12;

    drawBackground();
    draw(ballX, ballY, ballSize, ballSize);
}

// main
void loop()
{
    if(!(digitalRead(leftUpBtn) == LOW &&
    digitalRead(leftDownBtn) == LOW &&
```



```
digitalRead(rightUpBtn) == LOW &&
digitalRead(rightDownBtn) == LOW))
{
    coverOld(leftPaddleX, oldLeftY, paddleWidth,
paddleHeight);
    coverOld(rightPaddleX, oldRightY, paddleWidth,
paddleHeight);

    updatePaddle(leftY, oldLeftY, leftDownBtn, leftUpBtn);
    updatePaddle(rightY, oldRightY, rightUpBtn,
rightDownBtn);

    draw(leftPaddleX, leftY, paddleWidth, paddleHeight);
    draw(rightPaddleX, rightY, paddleWidth, paddleHeight);
}

coverOld(oldBallX, oldBallY, ballSize, ballSize);
draw(ballX, ballY, ballSize, ballSize);
moveBall();

delay(33);
}

// -----
void endScreen()
{
    screen.background(0, 0, 0);
    screen.stroke(255, 255, 255);
    screen.setTextSize(2);
```




```
screen.text("Game over", width / 6, height / 3);
screen.setTextSize(1);
screen.text("Restarting in 3..", width / 5 - 10, height / 3 + 25);
delay(1000);
screen.text("Restarting in 3..2..", width / 5 - 10, height / 3 +
25);
delay(1000);
screen.text("Restarting in 3..2..1", width / 5 - 10, height / 3 +
25);
delay(1000);
screen.noStroke();
screen.background(0, 0, 0);
drawBackground();
}
```

```
void drawBackground()
{
  screen.stroke(100, 100, 100);
  screen.fill(0, 0, 0);
  screen.circle(width / 2, height / 2, 37);
  screen.noStroke();

  screen.fill(100, 100, 100);
  screen.rect(width / 2, 0, 1, height);

  screen.fill(100, 100, 100);
  screen.circle(width / 2, height / 2, 3);

  // paddles and ball
  draw(leftPaddleX, leftY, paddleWidth, paddleHeight);
}
```




```
draw(rightPaddleX, rightY, paddleWidth, paddleHeight);
}

void updatePaddle(int& currentY, int& oldY, const int&
downBtn, const int& upBtn)
{
    oldY = currentY;
    if(digitalRead(downBtn) == HIGH && digitalRead(upBtn) ==
HIGH) {
        return;
    }

    if(digitalRead(downBtn) == HIGH)
    {
        // reached the bottom
        if(currentY + paddleHeight > height) {
            currentY = height - paddleHeight;
        }
        else{
            currentY += paddleJump;
        }
    }

    if(digitalRead(upBtn) == HIGH)
    {
        if(currentY - paddleJump < 0) {
            currentY = 0;
        }
        else {
            currentY -= paddleJump;
        }
    }
}
```



```
}  
}  
}
```

```
void coverOld(int oldX, int oldY, int width, int height)  
{  
    screen.fill(0, 0, 0);  
    screen.rect(oldX, oldY, width, height);  
}
```

```
void draw(int currentX, int currentY, int width, int height)  
{  
    screen.fill(100, 100, 100);  
    screen.rect(currentX, currentY, width, height);  
}
```

```
void moveBall()  
{  
    if(ballX == 0 || ballX + ballSize == width)  
    {  
        delay(500);  
        draw(ballX, ballY, ballSize, ballSize);  
        endScreen();  
        reset();  
        return;  
    }  
}
```

```
// ball movement  
oldBallX = ballX;  
oldBallY = ballY;
```



```
ballX += ballXDirection;  
ballY += ballYDirection;
```

```
validateBallPosition();
```

```
if(collideWithLeft() || collideWithRight())  
{  
    ballXDirection *= -1;  
    ballYDirection--;  
    return;  
}
```

```
if(ballX == 0 || ballX >= width - ballSize) // can be == ig  
{  
    bounceFromSide();  
}
```

```
if(ballY == 0 || ballY >= height - ballSize)  
{  
    bounceFromCeilingOrFloor();  
}  
}
```

```
void reset()  
{  
    ballX = width / 2;  
    ballY = height / 2;  
    ballXDirection = defaultSpeed;  
    ballYDirection = defaultSpeed;  
}
```



```
bool collideWithLeft()
{
    return ballX == leftPaddleX + ballSize && leftY <= ballY &&
    ballY <= leftY + paddleHeight;
}
```

```
bool collideWithRight()
{
    return ballX + ballSize == rightPaddleX && rightY <= ballY +
    ballSize && ballY + ballSize <= rightY + paddleHeight;
}
```

```
void bounceFromSide()
{
    ballXDirection *= -1;
}
```

```
void bounceFromCeilingOrFloor()
{
    ballYDirection *= -1;
}
```

```
void validateBallPosition()
{
    if(ballX < 0)
    {
        ballX = 0;
    }
}
```



```
if(ballY < 0)
{
    ballY = 0;
}

if(ballX > width)
{
    ballX = width;
}

if(ballY > height)
{
    ballY = height;
}
}
```



СОФИЙСКИ УНИВЕРСИТЕТ “СВ. КЛИМЕНТ ОХРИДСКИ”