МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Компьютерные технологии в проектировании и производстве»

Курсовая работа

«Игра Гексагон»

ОТЧЕТ

по курсовой работе

по дисциплине

Технологии программирования

РУКОВОДИТЕЛЬ:

Капранов С.Н.

Багиров М.Б.

СТУДЕНТ:

Докукин Д.В.

Работа защищена « »

С оценкой

Нижний Новгород

Содержание

1. Задание
2. Правила игры
3. Демонстрация
4. Программная реализация

1. Задание

Написать программу, играющую в Гексагон.

2. Правила игры

Поле состоит из 6-гранных клеток. Размер поля – по выбору.

Два игрока ходят по очереди. Возможны ходы:

1) Поставить новую свою шашку на любую свободную из 6 клеток, смежных со своей шашкой.

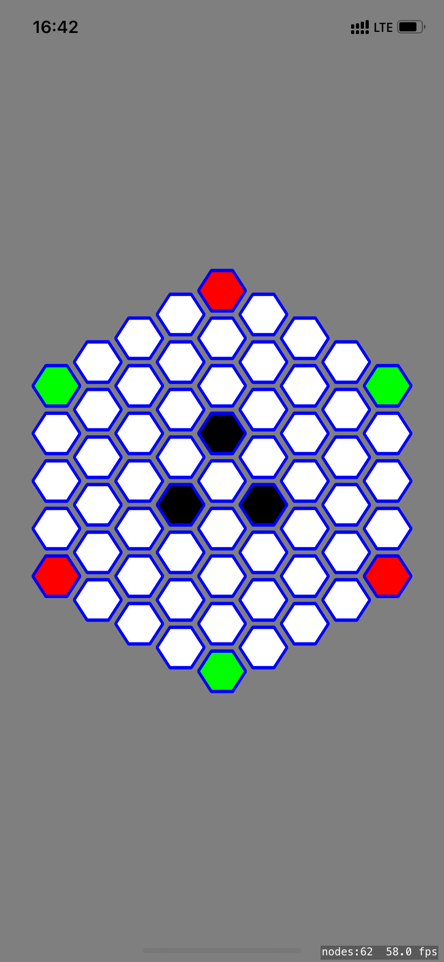
2) Переставить свою шашку на любую свободную клетку, смежную с клетками, смежными с прежним положением этой шашки.

В любом случае, если на клетках, смежных с новым положением своей шашки, есть шашки противника, они заменяются на свои.

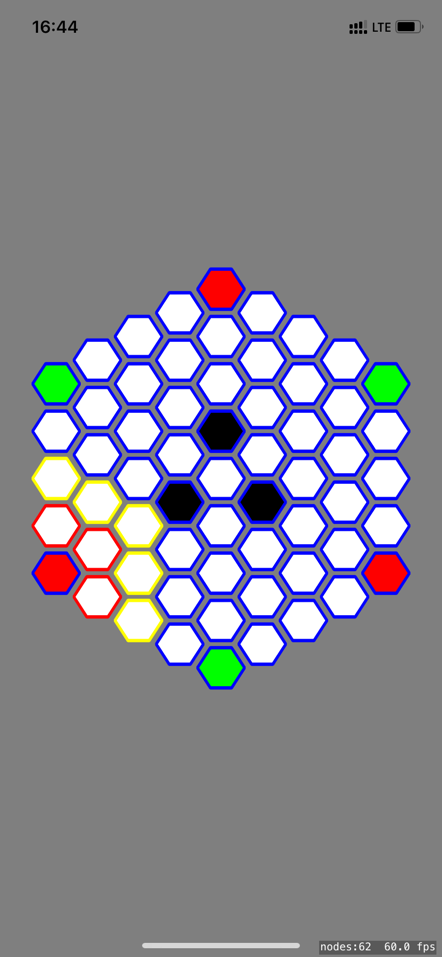
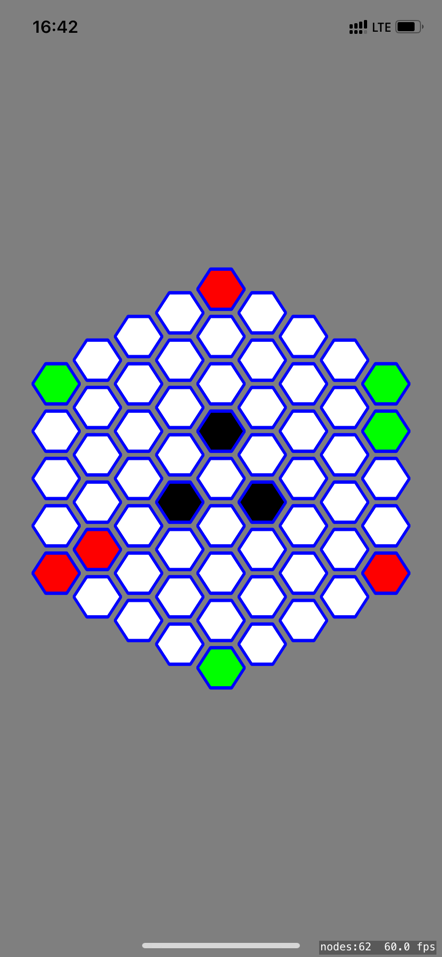
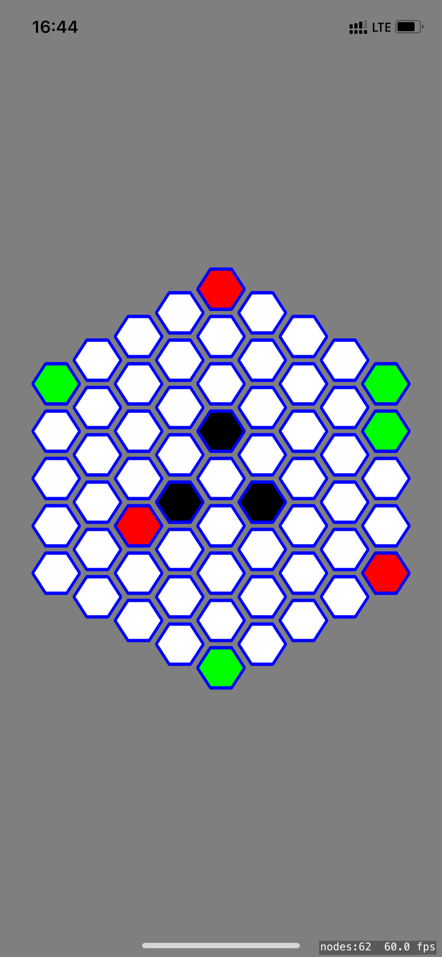
Игра заканчивается, когда все клетки заняты. Выигрывает тот, кто занял больше клеток.

3. Демонстрация

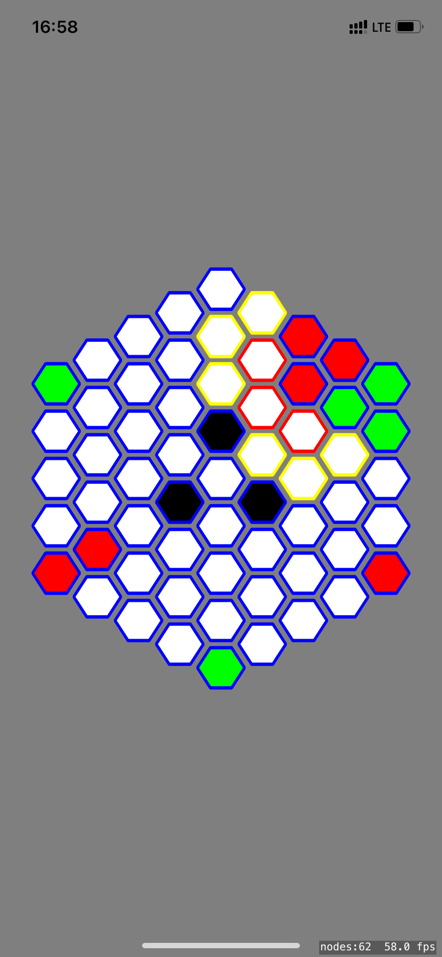
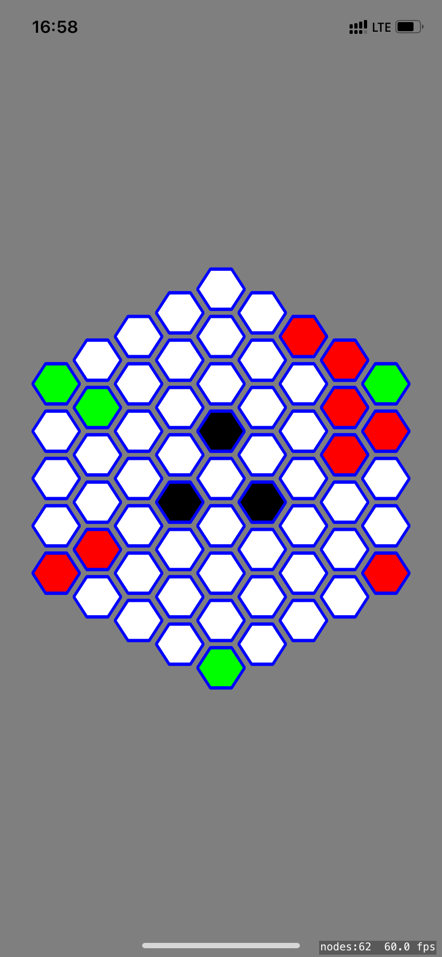
В приложении представлено два окна – стартовое и игровое.

По углам шестигранника размещаются шашки и чередуются по цвету. Игрок управляет красными шашками, программа – зелёными. Первым ходит игрок. Чтобы сделать ход, нужно нажать на любую из шашек, окруженную белыми клетками. Контур соседних клеток изменит цвет – красным будут отмечены клетки, в которые можно добавить новую шашку, а желтым – клетки, в которые выбранную шашку можно переместить. После нажатия на выбранную клетку, ход завершается.

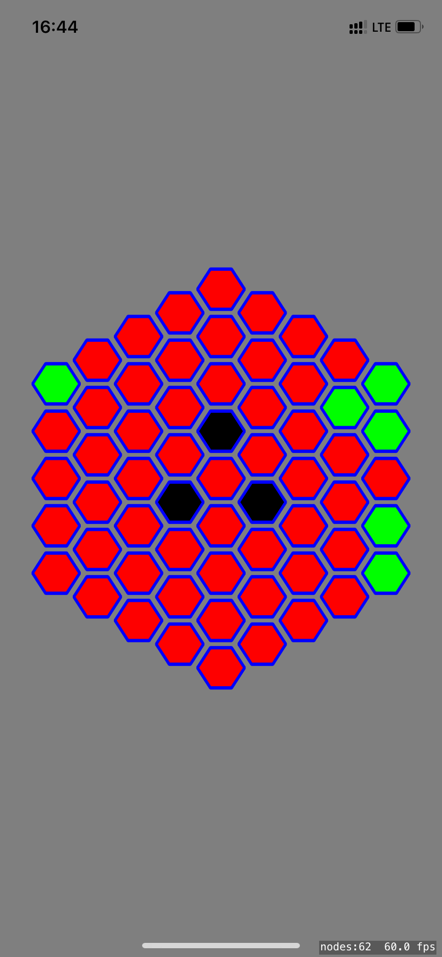
  

При приближении шашки вплотную к шашкам противника, они перекрашиваются.

Черные поля недоступны для размещения фишек и представляют собой преграды.

Игра заканчивается, когда у противника не остаётся возможности для хода. Побеждает сторона с большим количеством закрашенных полей.



4. Программная реализация

import SpriteKit

import GameplayKit

import UIKit

class GameScene: SKScene {

var StrMax: String = ""

var index = 0

var additionalDictionary: [Int: (i: Int, j: Int)] = [:]

var secondDictionary: [[Int]: Int] = [:]

var blackArray: [Int] = []

var ancorsArray = [SKShapeNode]()

var points = [CGPoint(x: -14, y: 25),

CGPoint(x: 14, y: 25),

CGPoint(x: 30, y: 0),

CGPoint(x: 14, y: -25),

CGPoint(x: -14, y: -25),

CGPoint(x: -30, y: 0),

CGPoint(x: -14, y: 25)

]

override func didMove(to view: SKView) {

SceneSetting()

createAncors()

preparingField()

}

func SceneSetting() {

self.backgroundColor = SKColor.gray

}

func createAncors() {

var x = 0, y = 240

for i in 1...9 {

x = 0 - 53\*(i-1)

y = 240 - 30\*(i-1)

for j in 1...9 {

if x >= 265 || x <= -265 {

x += 53

y -= 30

continue

}

ancorsArray.append(SKShapeNode(points: &points, count: points.count))

ancorsArray[index].lineWidth = 5 // задаем размер линий

ancorsArray[index].fillColor = SKColor.white

ancorsArray[index].strokeColor = SKColor.blue // задаем цвет контура

ancorsArray[index].position = CGPoint(x: x, y: y)

self.addChild(ancorsArray[index])

ancorsArray[index].name = String(index)

additionalDictionary[index]=(i: i, j: j)

//print(additionalDictionary[index] as Any)

secondDictionary[[i, j]] = index

x += 53

y -= 30

index += 1

}

}

print("The last index: = \(index-1)")

}

func strokeColoring(index internalIndex: Int) {

stepOne(I: -1, J: 0, externalIndex: internalIndex, color: .red, repeating: 1)

stepOne(I: 0, J: 1, externalIndex: internalIndex, color: .red, repeating: 1)

stepOne(I: 1, J: 1, externalIndex: internalIndex, color: .red, repeating: 1)

stepOne(I: 1, J: 0, externalIndex: internalIndex, color: .red, repeating: 1)

stepOne(I: 0, J: -1, externalIndex: internalIndex, color: .red, repeating: 1)

stepOne(I: -1, J: -1, externalIndex: internalIndex, color: .red, repeating: 1)

}

func stepOne(I: Int, J: Int, externalIndex: Int, color: SKColor, repeating: Int) {

let i = additionalDictionary[externalIndex]!.i + I

let j = additionalDictionary[externalIndex]!.j + J

if let secondIndex = secondDictionary[[i, j]] {

for a in 0...2 {

if secondIndex == blackArray[a] {return}

}

if ancorsArray[secondIndex].fillColor == SKColor.green {return}

if ancorsArray[secondIndex].strokeColor == SKColor.red { return}

if ancorsArray[secondIndex].fillColor == SKColor.red { return}

if ancorsArray[secondIndex].fillColor != SKColor.red {

ancorsArray[secondIndex].strokeColor = color

}

} else { return}

if repeating > 0 {

stepOne(I: -1, J: 0, externalIndex: secondDictionary[[i, j]]!, color: .yellow, repeating: repeating-1)

stepOne(I: 0, J: 1, externalIndex: secondDictionary[[i, j]]!, color: .yellow, repeating: repeating-1)

stepOne(I: 1, J: 1, externalIndex: secondDictionary[[i, j]]!, color: .yellow, repeating: repeating-1)

stepOne(I: 1, J: 0, externalIndex: secondDictionary[[i, j]]!, color: .yellow, repeating: repeating-1)

stepOne(I: 0, J: -1, externalIndex: secondDictionary[[i, j]]!, color: .yellow, repeating: repeating-1)

stepOne(I: -1, J: -1, externalIndex: secondDictionary[[i, j]]!, color: .yellow, repeating: repeating-1)

}

}

func paintStrokesBlue() {

for i in 0..<index {

ancorsArray[i].strokeColor = SKColor.blue

}

}

func preparingField() {

var a = secondDictionary[[1, 1]]!

ancorsArray[a].fillColor = SKColor.red

a = secondDictionary[[9, 5]]!

ancorsArray[a].fillColor = SKColor.red

a = secondDictionary[[5, 9]]!

ancorsArray[a].fillColor = SKColor.red

a = secondDictionary[[9, 9]]!

ancorsArray[a].fillColor = SKColor.green

a = secondDictionary[[1, 5]]!

ancorsArray[a].fillColor = SKColor.green

a = secondDictionary[[5, 1]]!

ancorsArray[a].fillColor = SKColor.green

a = secondDictionary[[4, 4]]!

blackArray.append(a)

ancorsArray[a].fillColor = SKColor.black

a = secondDictionary[[5, 6]]!

blackArray.append(a)

ancorsArray[a].fillColor = SKColor.black

a = secondDictionary[[6, 5]]!

blackArray.append(a)

ancorsArray[a].fillColor = SKColor.black

}

//var arrayOfRobotsCells: [Int] = []

var arrayWithGreenCells: [Int] = []

var additionalArray: [Int] = []

var copyFirstCircleCells: [Int] = []

var firstCircleCells: [Int] = []

var secondCircleCells: [Int] = []

var thirdCircleCells: [Int] = []

func robotsStep () {

arrayWithGreenCells = []

for n in 0..<ancorsArray.count {

if ancorsArray[n].fillColor == SKColor.green {

arrayWithGreenCells.append(n)

}

}

var enemInd: [Int]? = nil

var greenPoint: [Int] = []

for n in 0..<arrayWithGreenCells.count {

if let enemInd1 = thirdCircle(externalIndex: arrayWithGreenCells[n]) {

enemInd?.append(enemInd1)

greenPoint.append(arrayWithGreenCells[n])

}

copyFirstCircleCells = []

firstCircleCells = []

secondCircleCells = []

thirdCircleCells = []

}

if enemInd != nil {

var random0 = 0

//copyFirstCircleCells = firstCircleCells

for \_ in 0... {

random0 = Int(arc4random\_uniform(UInt32(greenPoint.count)))

firstCircle(externalIndex: greenPoint[random0])

if firstCircleCells.count == 0 {continue}

copyFirstCircleCells = firstCircleCells

firstCircle(externalIndex: enemInd![random0])

if firstCircleCells.count == 0 {

copyFirstCircleCells = []

firstCircleCells = []

continue}

}

for n in 0..<firstCircleCells.count {

for m in 0..<copyFirstCircleCells.count {

if firstCircleCells[n] == copyFirstCircleCells[m] {

ancorsArray[copyFirstCircleCells[n]].fillColor = SKColor.green

paintEnemy(externalIndex: copyFirstCircleCells[n], color: .green)

copyFirstCircleCells = []

firstCircleCells = []

secondCircleCells = []

thirdCircleCells = []

return

}

}

}

for n in 0..<firstCircleCells.count {

for m in 0..<secondCircleCells.count {

if firstCircleCells[n] == secondCircleCells[m] {

ancorsArray[copyFirstCircleCells[n]].fillColor = SKColor.green

paintEnemy(externalIndex: copyFirstCircleCells[n], color: .green)

copyFirstCircleCells = []

firstCircleCells = []

secondCircleCells = []

thirdCircleCells = []

return

}

}

}

} else {

headLoop: for n in 0..<arrayWithGreenCells.count {

//let rando = Int(arc4random\_uniform(UInt32(arrayWithGreenCells.count)))

firstCircle(externalIndex: arrayWithGreenCells[n])

if firstCircleCells.count > 0 {

let rand0 = Int(arc4random\_uniform(UInt32(firstCircleCells.count)))

ancorsArray[firstCircleCells[rand0]].fillColor = SKColor.green

paintEnemy(externalIndex: firstCircleCells[rand0], color: .green)

break headLoop

}

firstCircleCells = []

}

}

firstCircleCells = []

secondCircleCells = []

thirdCircleCells = []

}

func firstCircle(externalIndex: Int) -> Int? {

var i = 0

var j = 0

var enemyIndex: Int? = nil

for addi in [-1, 0, 1] {

preMainLoop: for addj in [-1, 0, 1] {

if (addi == 0 && addj == 0) || (addi == 1 && addj == -1) || (addi == -1 && addj == 1) {continue}

i = additionalDictionary[externalIndex]!.i + addi

j = additionalDictionary[externalIndex]!.j + addj

if let secondIndex = secondDictionary[[i, j]] {

for a in 0...2 {

if secondIndex == blackArray[a] {continue preMainLoop}

}

if ancorsArray[secondIndex].fillColor == SKColor.red {

enemyIndex = secondIndex

continue preMainLoop

} else if ancorsArray[secondIndex].fillColor == SKColor.green {

continue preMainLoop

}

firstCircleCells.append(secondIndex)

}

}

}

return enemyIndex

}

func secondCircle(externalIndex: Int) -> Int? {

var i = 0, j = 0

var enemyIndex: Int? = nil

for addi in -2...2 {

preMainLoop: for addj in -2...2 {

if (addi == 1 && addj == -2) || (addi == 2 && addj == -1) || (addi == 0 && addj == 0) {continue}

if (addi == -1 && addj == 2) || (addi == -2 && addj == 1) {continue}

if (addi == -2 && addj == 2) || (addi == 2 && addj == -2) {continue}

i = additionalDictionary[externalIndex]!.i + addi

j = additionalDictionary[externalIndex]!.j + addj

if let secondIndex = secondDictionary[[i, j]] {

for a in 0...2 {

if secondIndex == blackArray[a] {continue preMainLoop}

}

for n in 0..<firstCircleCells.count {

if secondIndex == firstCircleCells[n] { continue preMainLoop}

}

if ancorsArray[secondIndex].fillColor == SKColor.red {

enemyIndex = secondIndex

continue preMainLoop

} else if ancorsArray[secondIndex].fillColor == SKColor.green {

continue preMainLoop

}

secondCircleCells.append(secondIndex)

}

}

}

return enemyIndex

}

func thirdCircle(externalIndex: Int) -> Int? {

var i = 0, j = 0

var enemyIndex: Int? = nil

for addi in -3...3 {

preMainLoop: for addj in -3...3 {

if (addi == 1 && addj == -3) || (addi == 2 && addj == -3) || (addi == 3 && addj == -3) {continue}

if (addi == 3 && addj == -2) || (addi == 2 && addj == -2) || (addi == 3 && addj == -1) {continue}

if (addi == -3 && addj == 1) || (addi == -3 && addj == 2) || (addi == -3 && addj == 3) || (addi == 0 && addj == 0) {continue}

i = additionalDictionary[externalIndex]!.i + addi

j = additionalDictionary[externalIndex]!.j + addj

if let secondIndex = secondDictionary[[i, j]] {

for a in 0...2 {

if secondIndex == blackArray[a] {continue preMainLoop}

}

for n in 0..<firstCircleCells.count {

if secondIndex == firstCircleCells[n] { continue preMainLoop}

}

for n in 0..<secondCircleCells.count {

if secondIndex == secondCircleCells[n] { continue preMainLoop}

}

if ancorsArray[secondIndex].fillColor == SKColor.red {

enemyIndex = secondIndex

continue preMainLoop

} else if ancorsArray[secondIndex].fillColor == SKColor.green {

continue preMainLoop

}

thirdCircleCells.append(secondIndex)

}

}

}

return enemyIndex

}

func checkingCells(externalIndex: Int) -> Int? {

//var i = -1, j = -1

var indx: Int? = nil

if let ind = firstCircle(externalIndex: externalIndex) {

indx = ind

print("Enemy detected on level 1")

} else if let ind = secondCircle(externalIndex: externalIndex) {

indx = ind

print("Enemy detected on level 2")

} else if let ind = thirdCircle(externalIndex: externalIndex) {

indx = ind

print("Enemy detected on level 3")

}

return indx

}

func paintEnemy(externalIndex: Int, color: SKColor) {

var i = 0

var j = 0

for addi in [-1, 0, 1] {

preMainLoop: for addj in [-1, 0, 1] {

if (addi == 0 && addj == 0) || (addi == 1 && addj == -1) || (addi == -1 && addj == 1) {continue}

i = additionalDictionary[externalIndex]!.i + addi

j = additionalDictionary[externalIndex]!.j + addj

if let secondIndex = secondDictionary[[i, j]] {

for a in 0...2 {

if secondIndex == blackArray[a] {continue preMainLoop}

}

if color == SKColor.green {

if ancorsArray[secondIndex].fillColor == SKColor.red {

ancorsArray[secondIndex].fillColor = color

continue preMainLoop

} else if ancorsArray[secondIndex].fillColor == SKColor.green {

continue preMainLoop

}

firstCircleCells.append(secondIndex)

} else if color == SKColor.red {

if ancorsArray[secondIndex].fillColor == SKColor.green {

ancorsArray[secondIndex].fillColor = color

continue preMainLoop

} else if ancorsArray[secondIndex].fillColor == SKColor.red {

continue preMainLoop

}

}

}

}

}

}

var indexCatcher = 0

override func touchesBegan(\_ touches: Set<UITouch>, with event: UIEvent?) {

let touch = touches.first

if let location = touch?.location(in: self) {

let nodesArray = self.nodes(at: location)

let keyWord: String = nodesArray.first?.name ?? ""

if let helpNum = Int(keyWord) {

if ancorsArray[helpNum].fillColor == SKColor.red {

paintStrokesBlue()

strokeColoring(index: helpNum)

indexCatcher = helpNum

} else if ancorsArray[helpNum].strokeColor == SKColor.red {

ancorsArray[helpNum].fillColor = SKColor.red

paintEnemy(externalIndex: helpNum, color: .red)

paintStrokesBlue()

robotsStep()

} else if ancorsArray[helpNum].strokeColor == SKColor.yellow {

ancorsArray[helpNum].fillColor = SKColor.red

paintEnemy(externalIndex: helpNum, color: .red)

ancorsArray[indexCatcher].fillColor = SKColor.white

paintStrokesBlue()

robotsStep()

indexCatcher = 0

}

}

}

}

}