Тема: «Разработка игрового приложения «Puzzles»»

Выполнил студент группы ИС-32 Ваньшев П.А
Руководитель УП: Грахов И.В

Цель учебной практики – разработка игрового приложения для Android

Объект исследования – мобильные игры

Для достижения поставленной цели в данном проекте были решены следующие задачи:

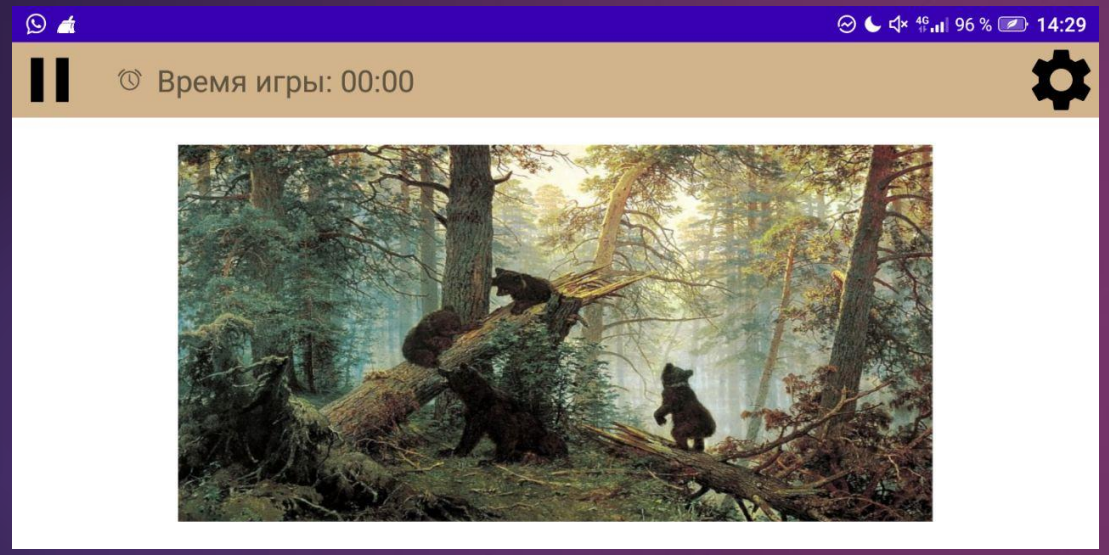
- Выбран жанр
- Определена тема игры
- Спроектированы структура и функциональность игры

Главное меню

На данный момент данное окно имеет функционал:

1) Возможность начать игру
2) Возможность открыть справку, для отображения информации автора

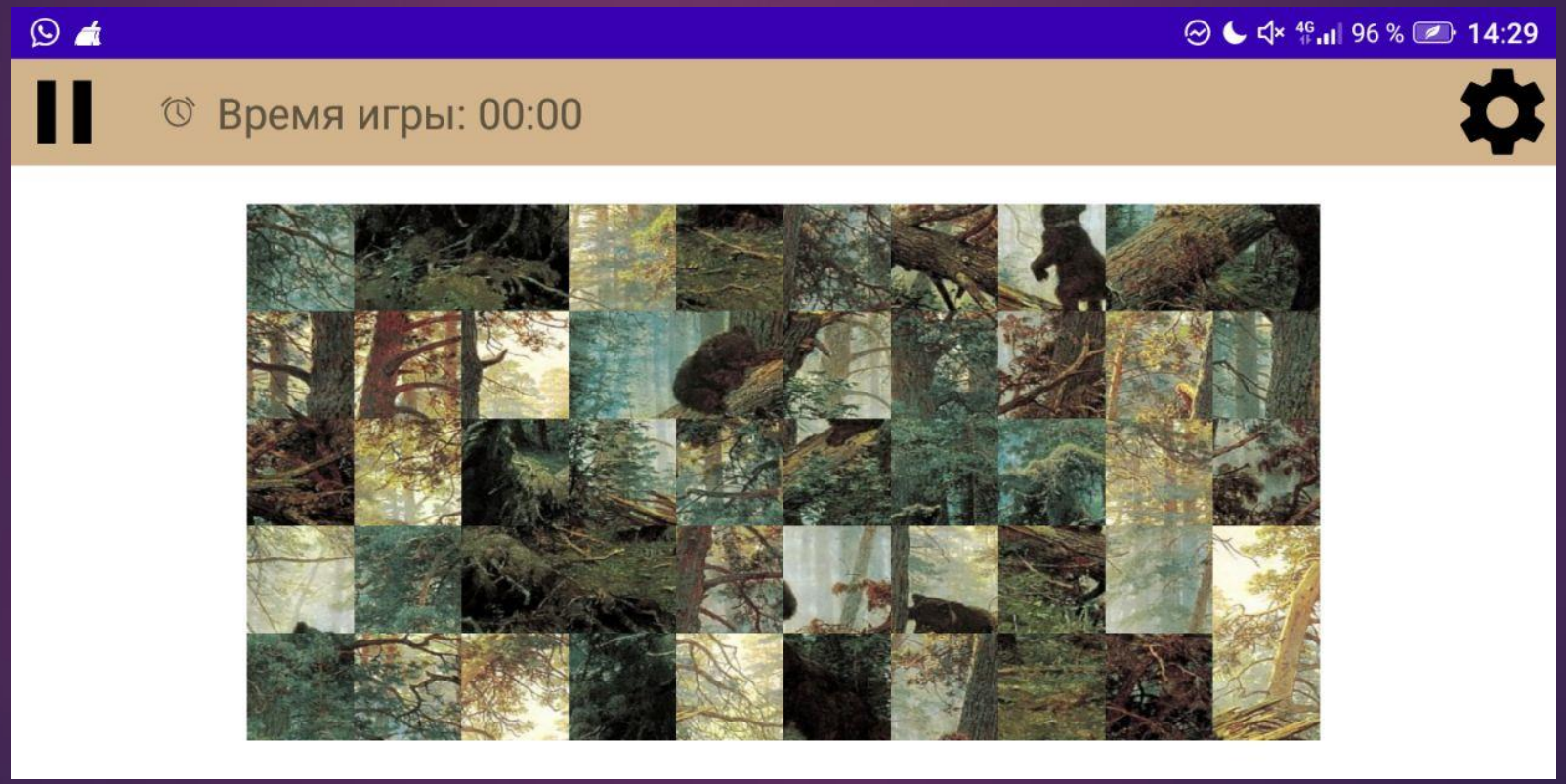
Игровой интерфейс



На данный момент данное окно имеет функционал:

1) Отображение изображения, которое должен собрать пользователь
2) Разбиение картинки на элементы

Перемешанное изображение



Код игрового поля

```
1  package com.example.puzzles
2
3  import android.content.pm.ActivityInfo
4  import androidx.appcompat.app.AppCompatActivity
5  import android.os.Bundle
6  import android.view.View
7  import android.widget.FrameLayout
8  import android.widget.ImageView
9  import android.widget.LinearLayout
10 import androidx.core.content.ContextCompat
11 import kotlinx.android.synthetic.main.activity_main.*
12
13 //Константы для формирования сетки игрового поля
14 const val ELEMENT_SIZE = 100
15 const val PLAYING_FIND_X = 10
16 const val PLAYING_FIND_Y = 5
17
18 class MainActivity : AppCompatActivity() {
19     private val allField = mutableListOf<Element>()
20     var tmpElementForTransferElement:Element? = null
21     private var indexElements:Int = 0 //Вспомогательный индекс
22
23     override fun onCreate(savedInstanceState: Bundle?) {
24         super.onCreate(savedInstanceState)
25         setContentView(R.layout.activity_main)
26         setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE)
27
28         //Создаётся элемент для отображения картинки-примера
29         val imageForGame = View(this)
30         imageForGame.layoutParams = LinearLayout.LayoutParams(ELEMENT_SIZE * PLAYING_FIND_X, ELEMENT_SIZE * PLAYING_FIND_Y)
31         imageForGame.background = ContextCompat.getDrawable(this, R.drawable.image_number_one)
32         playingField.addView(imageForGame)
```

```
34          //Устанавливается обработчик нажатия на примерочное изображение, для его разбеения на кусочки
35          playingField.setOnClickListener{
36              imageForGame.background = ContextCompat.getDrawable(this, R.color.teal_700)
37              for(index in (0 until PLAYING_FIND_X * PLAYING_FIND_Y)){
38                  randomPositionElement(indexElements)
39                  indexElements++
40              }
41              textView1.text = allField.size.toString()
42          }
43      }
44      //Метод необходим для корректной работы метода randomPositionElement(), для предотвращения пересечений объектов
45      private fun checkIfElementRepeatMargin(element: Element):Boolean{
46          if(allField.size > 0) {
47              for (index in (0 until allField.size))
48                  if(element.top == allField[index].top && element.left == allField[index].left){
49                      return false
50                  }
51                  else continue
52          }
53          return true
54      }
55
56      //Метод для генерации случайной позиции элемента
57      private fun randomPositionElement(indexElemnt:Int){
58          val element:Element = Element((0 until PLAYING_FIND_Y).random() * ELEMENT_SIZE,
59              (0 until PLAYING_FIND_X).random() * ELEMENT_SIZE,
60              null,
61              indexElemnt)
62          checkIfElementRepeatMargin(element)
63          addElementOfAllFieldArray(element.top, element.left, element.indexElement)
64      }
```

```
66      //Метод добавляет элемент(Кусочек) в массив, формирующий игровое поле
67      private fun addElementOfAllFieldArray(marginTop:Int, marginLeft:Int, index:Int){
68          var elementItem : Element = Element(marginTop, marginLeft, null, index)
69          if(checkIfElementRepeatMargin(elementItem)) {
70              val elementImage :ImageView = createImage(index)
71              elementItem = Element(marginTop, marginLeft, elementImage, index)
72              drawElementOfField(elementItem)
73              elementItem.imageView?.setOnClickListener{
74                  when (tmpElementForTransferElement) {
75                      null -> {
76                          tmpElementForTransferElement = elementItem
77                      }
78                      elementItem -> {
79                          //В дальнейшем будет возможность поворота элемента
80                      }
81                      else -> {
82                          clearViewArray(allField)
83                          allField[tmpElementForTransferElement!!.indexElement].top = elementItem.top
84                          allField[tmpElementForTransferElement!!.indexElement].left = elementItem.left
85                          allField[elementItem.indexElement].top = tmpElementForTransferElement!!.top
86                          allField[elementItem.indexElement].left = tmpElementForTransferElement!!.left
87                          drawAllArray(allField)
88                          tmpElementForTransferElement = null
89                      }
90                  }
91              }
92              allField.add(elementItem)
93          }
94          else{
95              randomPositionElement(index)
96          }
97      }
```

```
99      //Метод для отчистки игрового поля, для дальнейшей перерисовки игрового поля
100     private fun clearViewArray(array:MutableList<Element>){
101         for(item in array){
102             playingField.removeView(item.imageView)
103         }
104     }
105
106     //Метод для отрисовки всех элементов игрового поля по средствам многократного вызова drawElementOfField
107     private fun drawAllArray(array:MutableList<Element>){
108         for(item in array){
109             drawElementOfField(item)
110         }
111     }
112
113     //Метод для отрисовки 1 элемента
114     private fun drawElementOfField(element: Element) {
115         if (element.imageView == null) {
116             element.imageView = createImage(element.indexElement)
117         }
118         (element.imageView!!.layoutParams as FrameLayout.LayoutParams).topMargin = element.top
119         (element.imageView!!.layoutParams as FrameLayout.LayoutParams).leftMargin = element.left
120         playingField.addView(element.imageView)
121     }
122
```

```
123     //Метод для установки изображения в нужный элемент
124     private fun createImage(indexElemnt: Int) : ImageView{
125         val elementImage = ImageView(this)
126         if(indexElemnt != -1) {
127             when (indexElemnt) {
128                 0 -> elementImage.setImageResource(R.drawable.image_one_fragment_1)
129                 1 -> elementImage.setImageResource(R.drawable.image_one_fragment_2)
130                 2 -> elementImage.setImageResource(R.drawable.image_one_fragment_3)
131                 3 -> elementImage.setImageResource(R.drawable.image_one_fragment_4)
132                 4 -> elementImage.setImageResource(R.drawable.image_one_fragment_5)
133                 5 -> elementImage.setImageResource(R.drawable.image_one_fragment_6)
134                 6 -> elementImage.setImageResource(R.drawable.image_one_fragment_7)
135                 7 -> elementImage.setImageResource(R.drawable.image_one_fragment_8)
136                 8 -> elementImage.setImageResource(R.drawable.image_one_fragment_9)
137                 9 -> elementImage.setImageResource(R.drawable.image_one_fragment_10)
138                 10 -> elementImage.setImageResource(R.drawable.image_one_fragment_11)
139                 11 -> elementImage.setImageResource(R.drawable.image_one_fragment_12)
140                 12 -> elementImage.setImageResource(R.drawable.image_one_fragment_13)
141                 13 -> elementImage.setImageResource(R.drawable.image_one_fragment_14)
142                 14 -> elementImage.setImageResource(R.drawable.image_one_fragment_15)
143                 15 -> elementImage.setImageResource(R.drawable.image_one_fragment_16)
144                 16 -> elementImage.setImageResource(R.drawable.image_one_fragment_17)
145                 17 -> elementImage.setImageResource(R.drawable.image_one_fragment_18)
146                 18 -> elementImage.setImageResource(R.drawable.image_one_fragment_19)
147                 19 -> elementImage.setImageResource(R.drawable.image_one_fragment_20)
148                 20 -> elementImage.setImageResource(R.drawable.image_one_fragment_21)
149                 21 -> elementImage.setImageResource(R.drawable.image_one_fragment_22)
150                 22 -> elementImage.setImageResource(R.drawable.image_one_fragment_23)
151                 23 -> elementImage.setImageResource(R.drawable.image_one_fragment_24)
152                 24 -> elementImage.setImageResource(R.drawable.image_one_fragment_25)
153                 25 -> elementImage.setImageResource(R.drawable.image_one_fragment_26)
154                 26 -> elementImage.setImageResource(R.drawable.image_one_fragment_27)
155                 27 -> elementImage.setImageResource(R.drawable.image_one_fragment_28)
```

```
156                 28 -> elementImage.setImageResource(R.drawable.image_one_fragment_29)
157                 29 -> elementImage.setImageResource(R.drawable.image_one_fragment_30)
158                 30 -> elementImage.setImageResource(R.drawable.image_one_fragment_31)
159                 31 -> elementImage.setImageResource(R.drawable.image_one_fragment_32)
160                 32 -> elementImage.setImageResource(R.drawable.image_one_fragment_33)
161                 33 -> elementImage.setImageResource(R.drawable.image_one_fragment_34)
162                 34 -> elementImage.setImageResource(R.drawable.image_one_fragment_35)
163                 35 -> elementImage.setImageResource(R.drawable.image_one_fragment_36)
164                 36 -> elementImage.setImageResource(R.drawable.image_one_fragment_37)
165                 37 -> elementImage.setImageResource(R.drawable.image_one_fragment_38)
166                 38 -> elementImage.setImageResource(R.drawable.image_one_fragment_39)
167                 39 -> elementImage.setImageResource(R.drawable.image_one_fragment_40)
168                 40 -> elementImage.setImageResource(R.drawable.image_one_fragment_41)
169                 41 -> elementImage.setImageResource(R.drawable.image_one_fragment_42)
170                 42 -> elementImage.setImageResource(R.drawable.image_one_fragment_43)
171                 43 -> elementImage.setImageResource(R.drawable.image_one_fragment_44)
172                 44 -> elementImage.setImageResource(R.drawable.image_one_fragment_45)
173                 45 -> elementImage.setImageResource(R.drawable.image_one_fragment_46)
174                 46 -> elementImage.setImageResource(R.drawable.image_one_fragment_47)
175                 47 -> elementImage.setImageResource(R.drawable.image_one_fragment_48)
176                 48 -> elementImage.setImageResource(R.drawable.image_one_fragment_49)
177                 49 -> elementImage.setImageResource(R.drawable.image_one_fragment_50)
178             }
179         }
180         elementImage.layoutParams = FrameLayout.LayoutParams(ELEMENT_SIZE, ELEMENT_SIZE)
181         return elementImage
182     }
183 }
```

Благодарю за внимание!!!