

Разработка игрового приложения «gameFairy»

Отчет о проектной работе по курсу «Разработка
приложений для мобильных ОС»

Елизавета Андреевна Кузнецова и Никита Константинович Клюшов

17 января 2022

Введение

Каждый хоть раз в своей жизни сталкивался с компьютерными играми. Кто-то в них играл раньше, кто-то играет сейчас, а кто-то создает эти самые компьютерные игры, в которые потом играют другие пользователи. В нашем проекте будет расписано создание игрового приложения «gameFairy», который является объектом нашей работы.

Цель и задачи проекта

Цель проекта: разработать игровое приложение «gameFairy» на языке Java.

Задачи проекта:

1. Разработать требования к приложению;
2. Разработать графический интерфейс пользователя;
3. Реализовать приложение в Android Studio;
4. Получить навыки по составлению документации, описывающей работу программы.

План разработки игры

Разработка следующих Active:

1. Active, создающий меню.
2. Active с логикой игры.

Для реализации игры использован язык программирования Java.

Menu

Active, создающее меню игры. В него входят:

- кнопка начала игры "New Game"
- кнопка выхода из игры "Exit"
- кнопка включения музыки
- кнопка выключения музыки
- поле с подсчетом рекорда

Музыка

Код главного метода onCreate: запуск музыки

```
mPlayer = MediaPlayer.create(this, R.raw.music1);
mPlayer.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mp) {
        stopPlay();
    }
});
play();
```

play() и stopPlay() - методы запуска и остановки проигрывания музыки.

Музыка

Код главного метода onCreate: кнопки запуска и остановки музыки

```
stopMusic.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        stopPlay();
    }
});  
  
startMusic.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        play();
    }
});
```

Кнопки New Game и Exit

Код главного метода onCreate:

button - кнопка начала игры exit - кнопка выхода из игры

```
button.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        startActivity(intent);  
    }  
});  
  
exit.setOnClickListener(view -> {  
    finishAndRemoveTask();  
});
```

Рекорд

Рекорд хранится в prefs к которому мы получаем доступ с помощью SharedPreferences. Если мы получили результат выше, чем предыдущий рекорд, то мы обновляем рекорд.

```
void CountPoints()
{
    SharedPreferences prefs = this.getSharedPreferences("myPrefsKey", Context.MODE_PRIVATE);
    highScore = prefs.getInt("score", 0);
    if (pointsInt > highScore) {
        highScore = pointsInt;
        prefs.edit().putInt("score", highScore).apply();
        Intent intent = new Intent(this, Menu.class);
        startActivity(intent);
        finish();
    }
}
```

Active с логикой игры. В него входят:

- таймер
- подсчет жизни игрока
- подсчет заработанных очков
- фея, которую нужно выбрать
- появляющиеся феи

Таймер

Отсчитывает полторы минуты от начала игры, после чего завершает игру, выводит высplывающее сообщение с результатом игры и выходит в главное меню.

```
new CountDownTimer(30000, 1000) {
    @Override
    public void onTick(long l) {
        timer.setText("" + l/60000 + " : " + ((l/1000)
            - (60 * (l/60000))));
    }
    @Override
    public void onFinish() {
        ...
    }
}.start();
```

Набранные очки

Очки при нажатии на правильную фею высчитываются в отдельном методе AddPoint(), в котором сразу же обновляются на экране.

```
void AddPoint()
{
    pointsInt++;

    points.setText("" + pointsInt);
}
```

Текущее здоровье игрока

Жизни рассчитываются в методе LoseHeart(). Если игрок нажимает на неправильную фею, вызывается метод LoseHeart(), в котором уменьшается значение здоровья и в соответствии с ним отображаются сердечки здоровья на экране. Если жизнь заканчивается, всплывает окно с результатом и переходит в главное меню.

```
void LoseHeart()
{
    Intent intent2 = new Intent(this, LableEnd.class);
    hearts--;
    switch (hearts) {
        case ...
        ...
        finishAndRemoveTask();
    }
}
```

Фея

Случайная фея, которую нужно выбрать.

В методе SelectNewFairy() выбирается случайная фея и выводится в рамочку на экран. На картинку с такой же феей на игровом поле нужно будет нажать.

```
void SelectNewFairy(Integer[] fairies, ImageView fairyToCatch
{
    Random random = new Random();
    nextFairy = random.nextInt(5);

    fairyToCatch.setImageResource(fairies[nextFairy]);
}
```

Феи на игровом поле

Появляющиеся на игровом поле феи, на которые нужно нажимать.

Метод CheckFairy() - проверяет, правильная ли фея была нажата.

Метод IsFairyOverlap() - проверяет по координатам, не пересекаются ли феи.

Метод MoveRandomly() - задает фее случайный размер и позицию на игровом поле.

Обработка нажатия на фею(пример с феей bloom).

Вызываются методы CheckFairy(), MoveRandomly() и SelectNewFairy().

```
bloom.setOnClickListener(view -> {
    CheckFairy("bloom", fairiesString);
    MoveRandomly(fairies, width, height);
    SelectNewFairy(fairiesInteger, fairyToCatch);
});
```

Реализация приложения

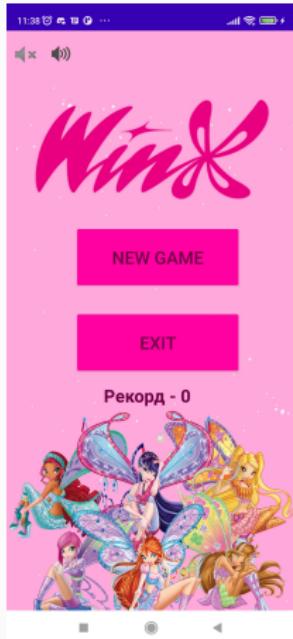


Рис. 1: Игра «gameFairy», главное меню

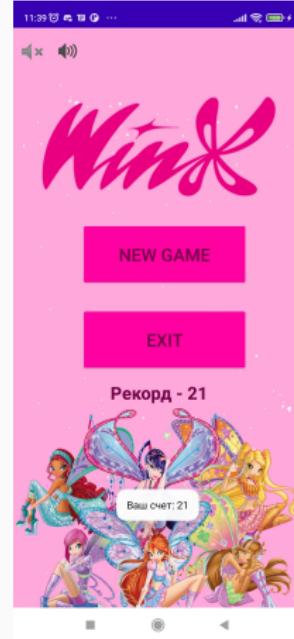


Рис. 2: Игра «gameFairy», главное меню после игры

Реализация приложения



Рис. 3: Игра «gameFairy», начало игры

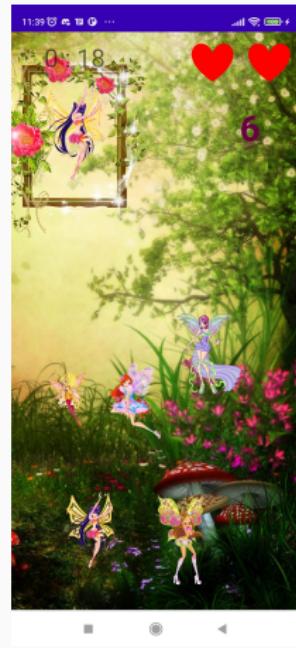


Рис. 4: Игра «gameFairy», в процессе игры

Заключение

Нами также был получен опыт работы со средой Android Studio, языком «Java» и его библиотеками, а также опыт создания мобильных приложений для платформы Android. Написание программы помогло закрепить теоретический материал на практике.

Игровое приложение «gameFairy» является логически завершенной игрой, но в дальнейшем возможны улучшения и добавления новых элементов и функций игры.

Спасибо за внимание !