WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA
z siedzibą w Rzeszowie

Programowanie urządzen mobilnych

Project

Cookie Clicker

Prowadzący: Dr. Marek Jaszuk

Authory:

Kiril Mosiichuk w68156

Kostiantyn Buchak w68136

Anton Hryshcenko w68343

Rzeszów 2024

Opis koncepcjii programy- Lista wymagań

VirusKillerClicker to aplikacja mobilna stworzona przy użyciu Android Studio oraz Kotlin, której celem jest umożliwienie użytkownikom atakowania wirusów poprzez klikanie, zdobywanie bonusów, oglądanie reklam nagradzających oraz przeglądanie sklepu z bonusami.

Główne wymagania aplikacji:

- **Ekran główny gry**:
  - Możliwość atakowania wirusów poprzez klikanie oraz długie klikanie.
  - Animacje wirusów w trakcie ataków.
- **System bonusów**:
  - Gromadzenie i wyświetlanie bonusów zdobytych podczas gry.
  - Możliwość przejścia do sklepu z bonusami.
- **Sklep z bonusami**:
  - Przeglądanie i zakup różnych bonusów przy użyciu zgromadzonych środków.
- **Reklamy**:
  - Wyświetlanie reklam nagradzających użytkowników za ich oglądanie.
  - Integracja z reklamami banerowymi oraz pełnoekranowymi.
- **Interfejs użytkownika**:
  - Wyświetlanie komunikatów Toast oraz Snackbar informujących użytkowników o wydarzeniach w grze.
  - Animowany interfejs użytkownika, poprawiający estetykę aplikacji.
- **Zarządzanie stanem gry**:

    - Przechowywanie i aktualizacja stanu gry przy użyciu ViewModel.
    - Obsługa stanów takich jak zapisywanie i ładowanie gry przy wstrzymaniu oraz wznowieniu aplikacji.
- **Nawigacja**:
    - Płynna nawigacja między ekranem głównym gry a sklepem z bonusami.

Dzięki spełnieniu tych wymagań, aplikacja **VirusKillerClicker** zapewni użytkownikom satysfakcjonującą rozgrywkę oraz intuicyjny interfejs użytkownika.

Opis implementacji – opis struktury programu z opisem roli każdego z elementów Aplikacja składa się z kilku głównych komponentów:

- **MainActivity.kt: Główny punkt wejścia aplikacji.**
- **GameFragment.kt: Zawiera główną logikę gry i interfejs użytkownika.**
- **Bonus.kt: Definiuje klasę danych Bonus.**
- **Bonuses.kt: Zawiera logikę związaną z kolekcjonowaniem i zarządzaniem bonusami.**
- **GameState.kt: Zarządza stanem gry.**
- **Virus.kt: Definiuje klasę danych Virus.**

Opis gry

Pierwszym, co widzi użytkownik po wejściu do programu jest głowny ekran ktory zawiera:

1. czaszka - liczba osadzonych wirusów
2. liczba zaangażowanych osób
3. równowaga
4. Poziom graczy
5. doświadczenie do następnego lvlu
6. Sklep

7. Ikona samego wirusa na którą trzeba klikać , ilość jego “hp”, jego “lvl” i reward za zabijenie virusa

Opis sklepu:

Przy wejściu do sklepu użytkownik widzi rożne rodzaje przedmiotów do kupienia które bedą pomagać jemy w trakcie gry

Kod Programu

```
class Virus()
{

    companion object
    {

        private val viruses = arrayOf(
            arrayOf(25, 50, R.drawable.virus_0),
            arrayOf(50, 110, R.drawable.virus_1),
            arrayOf(100, 220, R.drawable.virus_2),
            arrayOf(170, 390, R.drawable.virus_3),
            arrayOf(270, 650, R.drawable.virus_4),
            arrayOf(400, 1000, R.drawable.virus_5),
            arrayOf(460, 1200, R.drawable.virus_6),
            arrayOf(620, 1670, R.drawable.virus_7),
            arrayOf(820, 2300, R.drawable.virus_8),
            arrayOf(1040, 3010, R.drawable.virus_9),
            arrayOf(1300, 3900, R.drawable.virus_10),
            arrayOf(1600, 4960, R.drawable.virus_11),
            arrayOf(1940, 6210, R.drawable.virus_12),
            arrayOf(2320, 7650, R.drawable.virus_13)
        )

        val maxLvl = viruses.size
    }


    private val _hpString = MutableLiveData<String>()
    val hpString: LiveData<String>
        get() = _hpString

    private val _rewardString = MutableLiveData<String>()
    val rewardString: LiveData<String>
        get() = _rewardString

    private val _lvlString = MutableLiveData<String>()
    val lvlString: LiveData<String>
        get() = _lvlString

    private var _lvl: Byte = 0
        set(value)
        {
```

MainActivity.kt | GameViewModel.kt | MenuFragment.kt | MenuViewModel.kt | Virus.kt | GameState.kt | App.kt | Bonus

```
    }


    fun attackViruses(longClick: Int = 1)
    {
        Log.i("Base attack per click: ${bonuses.numbersAttackPerClickValue}. Critical Attack: ${bonuses.criticalAttackValue}. Bonus attack mult

        var dmg = 0
        for (i in 1..((bonuses.numbersAttackPerClickValue * bonusAttackMultiplier) * longClick))
        {
            dmg += if ((1..100).random() < bonuses.criticalAttackValue) // crit
            {
                Log.i("Crit")
                2
            }
            else
            {
                Log.i("Normal")
                1
            }
        }

        Log.i("$dmg dmg total")
        _dmg.value = dmg.toString()

        if (virus.attackVirus(dmg))//virus dead
        {
            Log.i("Dead")
            _money.value = _money.value!!.plus(
                (virus.reward.times(_bonuses.rewardMultiplierValue).toInt())
            )

            _xp = _xp.plus((virus.reward).div(10))

            if (_lvl.value!! < maxLvl && _xp >= _xpToNextLvl) //lvl upgrade
            {
                Log.i("New Lvl")
                _lvl.value = _lvl.value!!.plus(1)
                _xpToNextLvl = if (_lvl.value!! != maxLvl)
                {
                    xpArray[_lvl.value!!]
```

```kotlin
class Bonuses(
    multiplier0: Byte,
    multiplier1: Byte,
    multiplier2: Byte,
    multiplier3: Byte,
    critAttack: Byte,
    coinsPM: Byte,
    store: Byte,
    rewardMulti: Byte,
    attackPC: Byte
)
{

    companion object
    {
        val pricesSLM =
            intArrayOf(100, 200, 400, 800, 1600, 3200, 6400, 12_800, 25_600, 51_200, 102_400)
        val valuesSLM = arrayOf(0, 1, 2, 4, 8, 16, 32, 64, 128, 252, 512, 1024)

        val pricesCA = intArrayOf(500, 2000, 5000, 12000, 28000, 56000)
        val valuesCA = arrayOf(1, 5, 10, 25, 50, 75, 100)

        val pricesNAPC = intArrayOf(10_000, 25_000, 35_000, 50_000)
        val valuesNAPC = arrayOf(1, 2, 3, 4, 5)

        val pricesCPM = intArrayOf(1200, 3600, 6000, 18_000, 30_000)
        val valuesCPM = arrayOf(1, 2, 5, 10, 25, 50)

        val pricesRM = intArrayOf(2_000, 5_000, 12_000, 30_000)
        val valuesRM = arrayOf(1F, 1.05F, 1.1F, 1.2F, 1.5F)

        val pricesS = intArrayOf(600, 1200, 3000, 15000, 60000, 120000, 240000)
        val valuesS = arrayOf(60, 120, 300, 1500, 3000, 6000, 12000, 24000)
    }

    //region bonuses
```



```kotlin
import com.myhiprojects.viruskiller.utils.BonusAdapter

class ShopFragment : Fragment()
{
    private lateinit var viewModel: ShopViewModel
    private lateinit var viewModelFactory: ShopViewModelFactory
    private lateinit var bonusAdapter: BonusAdapter
    private lateinit var binding: FragmentShopBinding

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View
    {
        val gson = Gson()
        val bonusesData = gson.fromJson(
            ShopFragmentArgs.fromBundle(requireArguments()).bonuses,
            BonusesData::class.java
        )

        binding = DataBindingUtil.inflate(
            inflater,
            R.layout.fragment_shop,
            container,
            false
        )

        viewModelFactory = ShopViewModelFactory(
            ShopFragmentArgs.fromBundle(requireArguments()).money,
            Bonuses(bonusesData),
            requireContext()
        )

        viewModel = ViewModelProvider(this, viewModelFactory).get(ShopViewModel::class.java)


        binding.shopViewModel = viewModel
        binding.lifecycleOwner = this
```

Podsumowanie

W tym projekcie stworzono grę mobilną za pomocą Android Studio, projektując i implementując różne ekrany aplikacji, takie jak ekran główny gry, sklep, oraz interfejs użytkownika związany z reklamami. Umożliwili użytkownikom atakowanie wirusów poprzez klikanie i długie klikanie, przeglądanie sklepów z bonusami oraz zdobywanie nagród za oglądanie reklam.

W szczególności, zaprojektowano i wdrożono:

- Animacje: Animacje wirusów dla krótkich i długich kliknięć, które zwiększają interaktywność gry.
- Reklamy nagradzające: Integrację z reklamami, które nagradzają użytkowników po ich obejrzeniu.
- Widok Model (ViewModel): Użycie architektury MVVM do zarządzania stanem gry i logiką.
- Komunikaty Toast i Snackbar: Wyświetlanie informacji i nagród za pomocą komunikatów Toast i Snackbar.
- Nawigacja: Nawigację między różnymi ekranami aplikacji, takimi jak przejście do sklepu.

Dzięki temu projektowi, nauczyliśmy się praktycznego zastosowania narzędzi Android Studio, programowania w Kotlinie, oraz tworzenia funkcjonalnych i estetycznych interfejsów użytkownika. Ponadto, zdobyliśmy doświadczenie w integracji reklam w aplikacjach mobilnych oraz zarządzaniu stanem gry za pomocą architektury MVVM.

Made as a Project for WSIiZ

Developed by:
w68136 Kostiantyn Buchak (functionality)
w68156 Kiril Mosiichuk (structure)
w68343 Anton Hryshchenko (textures and design)

Language used: Kotlin

Guides/sources used:
ADs https://www.youtube.com/watch?v=NXuXbXN1llY
Gradient background https://www.youtube.com/watch?v=GxKqln50jsw
Animations https://www.youtube.com/watch?v=_P_Z5wIxGOc
Activity switch https://www.youtube.com/watch?v=2gljhNFKimk

Link do projectu

https://github.com/antonada/AndroidApp