

AKADEMIA NAUK STOSOWANYCH W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Operatsiya 'Kooperatsiya' (Operacja kooperacja)

Autor:
Maciej Śmierciak
Michał Jonak
Konrad Szczurek

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2022

Spis treści

1. Ogólne określenie wymagań	5
1.1. Gra logiczna	5
1.1.1. Tryb Graficzny	8
1.1.2. Tryb Tekstowy	9
2. Określenie wymagań szczegółowych	12
2.1. Założenia główne	12
2.1.1. Utrzymanie modułowości projektu	12
2.1.2. Łatwość implementacji	12
2.1.3. Prostota w testowaniu i ewentualnym debuggingu	12
2.1.4. Podzielenie aplikacji na 2 tryby	12
2.1.5. Użycie latarki	13
2.2. Struktura aplikacji	13
2.2.1. Menu główne	13
2.2.2. Menu ustawień	13
2.2.3. Dwa tryby gry	14
3. Projektowanie	15
3.1. Wykorzystane narzędzia	15
3.1.1. Xamarin	15
3.1.2. Xamarin Community Toolkit	15
3.1.3. Microsoft Visual Studio	16
3.1.4. Git, GitHub	17
3.1.5. Język C#	18
3.1.6. Baza danych SQLite	19
3.2. Działanie aplikacji	20
3.2.1. Menu główne	20
3.2.2. Ustawienia	22
3.2.3. Tryb graficzny	23
4. Implementacja	28

4.1. Przycisk	28
4.1.1. Użyte atrybuty przycisku (listing 1)	29
4.1.2. Inne użyteczne atrybuty	29
4.1.3. Przyciski w formie zdjęcia	30
4.1.4. Użyte atrybuty	30
4.2. MainPage	31
4.3. OnAppearing	31
4.4. Animate_pulse	32
4.5. FirstTask	33
4.6. Wciśnięcie złego przycisku	33
4.7. HUD	34
4.8. Literaki	35
4.9. SetTime	36
4.10. SetColumns	36
4.11. RandomizeWordTable	37
4.12. RandomizeText	38
4.13. MoveUp	38
4.14. ChangeBlockText	39
4.15. CheckIfWin	39
4.16. MoveDown	40
5. Testowanie	42
6. Podręcznik użytkownika	46
6.1. Menu główne	46
6.2. Tryb tekstowy	48
6.3. Labirynt	48
6.4. Literaki	51
6.5. Kod Morsa	54
6.6. Kolorki	57
6.7. Wielki Przycisk	60
6.8. Tabela wyników	63
Literatura	66

Spis rysunków	67
Spis tabel	68
Spis listingów	69

1. Ogólne określenie wymagań

1.1. Gra logiczna

Projektem jest gra logiczna możliwa do zagrań tylko w trybie kooperacji. Gra będzie opierać się na stosunkowo łatwych zagadkach, które będzie można rozwiązać tylko współpracując.

Ogólnym konceptem jest podzielenie gry na 2 główne części:

- Tryb graficzny,
- Tryb tekstowy.



Rys. 1.1. Labirynt

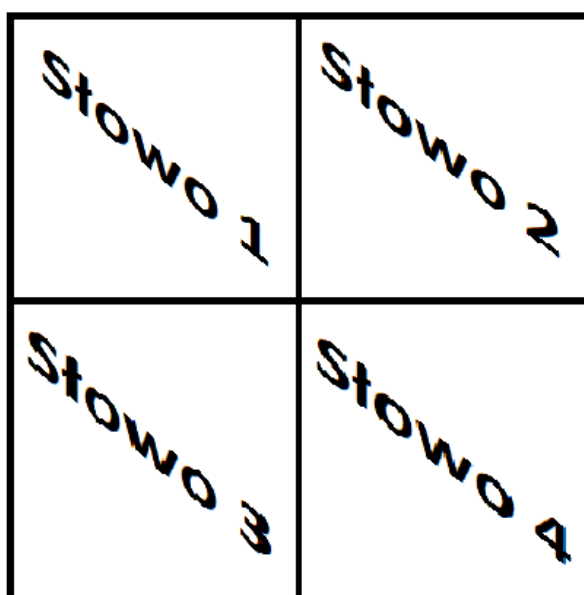
Jak możemy zauważyć na rysunku 1.1 będziemy mieli określoną liczbę żyć na rozwiązanie określonej liczby zagadek w określonym czasie. W tym trybie zagadki będą polegały na wyjściu z labiryntu.



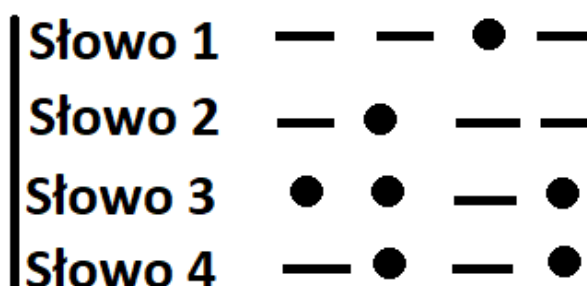
Rys. 1.2. Literaki

W trybie gry Literaki gracze mają za zadanie ułożyć czteroliterowy wyraz, który nakłada się z wyrazem w bazie. Dostęp do bazy wyrazów ma gracz w trybie tekstowym. Gracz obsługujący tryb graficzny za pomocą strzałek zmienia litery na danej pozycji.

Ze względu na to, że nie wszystkie litery są na wszystkich polach możliwość będzie tylko jedna. Błędna kombinacja oznacza utratę jednego z żyć. Podobnie jak w trybie labiryntu po utracie 3 żyć gracze przegrywają.



Rys. 1.3. Kod - Tryb graficzny



Rys. 1.4. Kod - Tryb tekstowy

Trzeci tryb gry, który zostanie zaimplementowany do gry będzie oparty na latarce zawartej w telefonie. Telefon osoby obsługującej tryb graficzny włączy i wyłączy latarkę określoną liczbę razy tworząc przy tym "kod morse" opisany w trybie tekstowym. Gracz obsługujący tryb graficzny będzie miał za zadanie zapamiętać stosunkowo krótki kod i podać go osobie będącej w trybie graficznym. Następnie osoba zarządzająca trybem graficznym musi dopasować go do jednego ze słów po czym podaje słowo kluczowe współnikowi. Wybór złego słowa pozbawia nas jednego życia i losuje nowy sygnał.

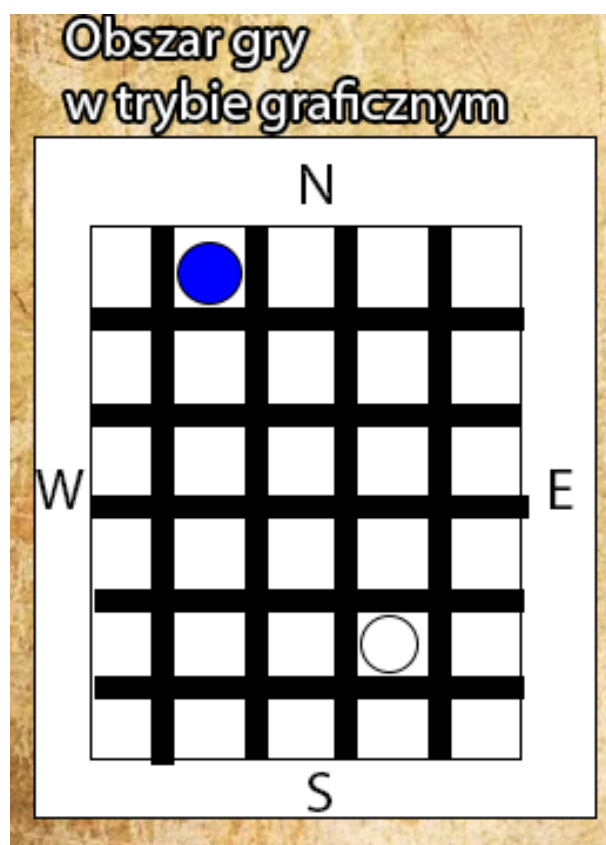
Kolejnym trybem gry będą "Kolorki". Będzie on polegał na odtworzeniu określonej sekwencji po jej wyświetleniu. Utrudnieniem będzie to, że każdy kolor będzie odpowiadał innemu, co będzie opisane w trybie graficznym. Gracz trybu graficznego

będzie miał za zadanie komunikowania partnerowi jaki kolor został wyróżniony po czym kliknięcie w jego odpowiednik opisany w trybie tekstowym.

Ostatnia zagadka będzie stosunkowo prosta, będzie ona polegała na przytrzymaniu przycisku przez określony czas wtedy kiedy timer jako ostatnią cyfrę będzie miał cyfrę przypisaną do określonego koloru przycisku. Wszystko co musi wtkonać gracz w trybie graficznym jest opisane w trybie tekstowym.

1.1.1. Tryb Graficzny

Będzie opierał się na rozwiązywaniu zagadek. Gracz sam nie będzie w stanie rozwiązać zagadki, ponieważ podpowiedzi czy też cała solucja danej zagadki będą zawarte w trybie tekstowym. W tym trybie będziemy widzieć plansze rozgrywki i będziemy mogli sterować naszą postacią.



Rys. 1.5. Labirynt

W trybie graficznym, jak widać na powyższym rysunku, widzimy naszą postać, niebieską kulkę, i nasz cel, białą kulkę. Natomiast nie widzimy drogi do mety i w tym

celu musimy komunikować się z partnerem.

Za każdym razem jak wykonamy zły ruch czyli wejdziemy w ścianę nasza kulka będzie wracać na początek trasy a my tracimy jedno z naszych żyć. Po utracie wszystkich żyć kończymy rozgrywkę.

Innym trybem gry będą literaki polegające na układaniu słów. Gracz w tym trybie będzie za pomocą strzałek zmieniał litery na określonych pozycjach. Błędna kombinacja prowadzi do utraty życia i jest równoznaczna z wejściem w ścianę w trybie labiryntu.

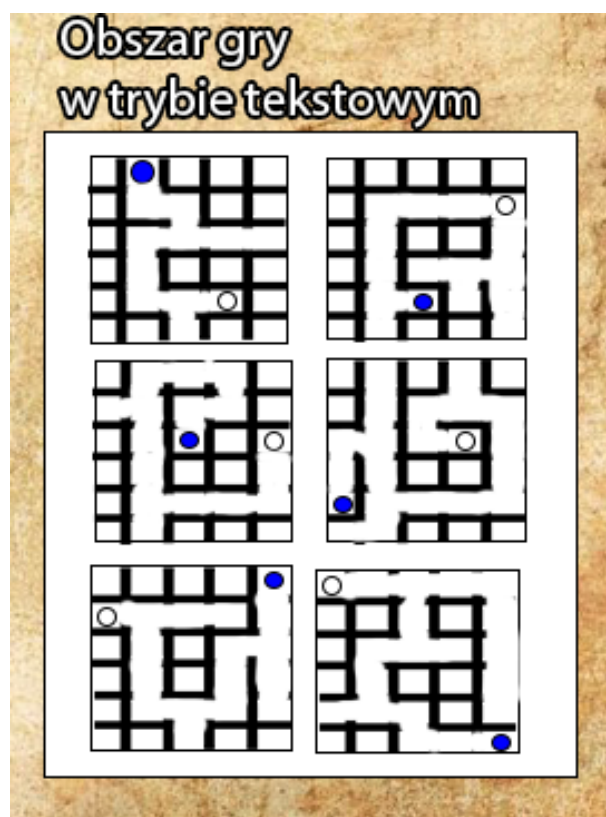
Tryb graficzny trzeciej gry będzie oparty na prostej tabeli z paroma różnymi opcjami do wyboru. W zależności od otrzymanych instrukcji od naszego współnika będziemy musieli wybrać jedną z nich. Telefon gracza obsługującego ten tryb będzie za pomocą latarki wyświetlał jeden z dwóch rodzajów sygnału. Będzie to kod oparty o kod morsa ale ze zmienionymi słowami. Dla przykładu jeżeli latarka włączy się raz na długo a potem 3 razy mrugnie będzie to oznaczało jeden sygnał długi (oznaczenie w trybie tekstowym - kreska) i 3 krótkie (w trybie tekstowym - kropka).

W trybie graficzny w czwartej zagadce będziemy musieli obserwować, który kolor zostaje wyróżniony. Następnie komunikujemy się z partnerem, który wskazuje nam odpowiadający kolor dla wyróżnionego po czym go klikamy. Sekwencja wydłuża się do pięciu wyróżnionych kolorów. Gra kończy się wraz ze zgadnięciem pięciu odpowiednich kolorów.

Ostatnia zagadka będzie stosunkowo prosta. Wymagane będzie od nas tylko przytrzymanie przycisku przez odpowiednio długi czas.

1.1.2. Tryb Tekstowy

Będzie opierał się na znajdowaniu podpowiedzi czy też solucji do aktualnie wykonywanej zagadki przez osobę obsługującą tryb graficzny. Naszym zadaniem będzie współpraca z osobą, która steruje postacią w trybie graficznym w celu jak najefektywniejszego ukończenia zagadek przed końcem ustalonego czasu.



Rys. 1.6. Labirynt

Jak można zobaczyć na rysunku 1.6 w trybie tekstowym będziemy widzieć dostępne mapy rozgrywki. Zadaniem gracza w trybie tekstowym będzie takie poprowadzenie partnera w trybie graficznym, żeby niebieska kula dotarła do mety (białej kuli) unikając wchodzenia w ściany.

W trybie gry "Literaki" gracz (obsługujący tryb tekstowy) będzie miał dostępną bazę ze słowami i będzie musiał dzięki komunikacji z graczem operującym interfejsem graficznym pomóc mu ułożyć pasujące słowo. Gracz w trybie graficznym będzie miał opcję ułożenia tylko jednego słowa z bazy.

W trzecim trybie gry będziemy musieli za pomocą komunikacji z naszym partnerem rozszyfrować o jaki kod chodzi w danym momencie. W trybie tekstowym będziemy otrzymywać informacje dotyczące "wyglądu kodu". W rzeczywistości otrzymamy informację ile było długich sygnałów a ile krótkich i w jakiej kolejności. Po otrzymaniu takowych informacji gracz będzie stawał przed wyborem jednego ze słów, które w tym przypadku będzie oznaczone przed chwilą otrzymanym kodem. Jeżeli żadne słowo się nie będzie zgadzało to najprawdopodobniej otrzymaliśmy złe odpowiedzi od naszego partnera. Jeżeli jednak znajdujemy pasujący kod to podajemy słowo,

które ten kod opisuje do operatora trybu graficznego.

W czwartej zagadce będziemy mieli opisane, który kolor odpowiada któremu. Dla przykładu jeżeli zostanie wyróżniony kolor zielony to naszym zadaniem będzie poszukiwanie, który kolor jest przypisany do niego (na przykład czerwony) i zakomunikowanie tego naszemu partnerowi.

W piątej zagadce będziemy mieli opis jaki kolor przycisku odpowiada ilości przytrzymanych sekund i jaka musi być ostatnia cyfra na timerze. Po sprawdzeniu tego przekazujemy naszemu partnerowi wszystkie te informacje a on kończy w ten sposób grę.

2. Określenie wymagań szczegółowych

2.1. Założenia główne

- Utrzymanie modułowości projektu
- Łatwość implementacji
- Prostota w testowaniu i ewentualnym debuggingu
- Podzielenie aplikacji na 2 tryby
- Użycie latarki

2.1.1. Utrzymanie modułowości projektu

Pozwoli to na pracę nad wieloma "poziomami" jednocześnie co przełoży się na lepsze rozłożenie pracy pomiędzy członków grupy. Ponadto pozwoli to na szybsze wykonywanie niektórych elementów aplikacji.

2.1.2. Łatwość implementacji

Pozwoli to na testowanie każdego modułu osobno. Dzięki temu rozwiązaniu będziemy mogli lepiej wyeliminować błędy a co za tym idzie lepiej dopracować nasz projekt.

2.1.3. Prostota w testowaniu i ewentualnym debuggingu

Chcemy dążyć do jak najłatwiejszego i jednocześnie najbardziej efektywnego sposobu testowania aplikacji. Pozwoli nam to zaoszczędzić cenny czas, który będziemy mogli poświęcić na lepsze dopracowanie szczegółów.

2.1.4. Podzielenie aplikacji na 2 tryby

Jednym z głównych rozwiązań w aplikacji będzie podzielenie jej na 2 zależne od siebie tryby (tryb graficzny i tekstory) zamiast tworzenia osobnej aplikacji dla każdego z trybów.

Jedną z wielu zalet tego rozwiązania będzie zmniejszenia nakładów pracy dzięki skupieniu się tylko na jednej aplikacji. Dzięki temu aplikacja będzie bardziej dopracowana pod względem działania czy też wyglądu.

Drugą zaletą będzie łatwiejsze przeszukiwanie aplikacji pod względem błędów, testowanie jej czy też naprawa potencjalnych błędów, ponieważ nie trzeba będzie naprawiać tego samego błędu niekiedy w obydwu aplikacjach.

Trzecią zaletą będzie łatwość korzystania z aplikacji - obydwaj gracze muszą posiadać tą samą grę a nie dwie różne wersje. Dzięki temu łatwiej będzie można zamienić się trybem gry z partnerem co może zwiększyć radość z gry.

2.1.5. Użycie latarki

Jedna z zagadek będzie oparta na wysyłaniu sygnałów w formie kodu morsa za pomocą latarki. Latarka będzie uruchamiana na określony odstęp czasu po czym zostanie wyłączona i włączona ponownie jeżeli ostatni sygnał nie został pokazany. Kod będzie oparty na 2 sygnałach: krótkim i długim.

2.2. Struktura aplikacji

W aplikacji będzie dostępne:

- Menu główne
- Menu ustawień
- Dwa tryby gry
 - Tryb graficzny
 - Tryb tekstowy

2.2.1. Menu główne

W tym panelu będziemy mieli dostęp zarówno do wyboru trybu gry jak i do ustawień. Ten panel będzie przejrzysty i łatwy w obsłudze, wszystkie opcje będą podpisane lub będą zawierały adekwatną do nazwy ikonę. Dla przykładu ustawienia zostaną oznaczone zębatką z podpisem ustawienia. Menu główne już na początku będzie zawierać jedną łatwą zagadkę.

2.2.2. Menu ustawień

Ten panel umożliwi użytkownikom, jak sama nazwa wskazuje, ustawienia rozgrywki takie jak głośność muzyki. Do palen ustawień będzie można wejść z każdego innego

panelu. Zmiana ustawień będzie działała globalnie czyli zmiana głośności poskutkuje zmianą głośności w każdym pozostałym panelu, do którego przejdziemy.

2.2.3. Dwa tryby gry

Użytkownicy będą mieli do wyboru tryb gry. Jeżeli użytkownik zdecyduje się na wybór trybu graficznego jedyne co będzie musiał zrobić to kliknąć w odpowiedni przycisk oznaczony jako tryb graficzny. Tryby gry będą podpisane i każdy z nich będzie miał osobny panel odpowiadający za określone funkcję, które będą potrzebne do rozwiązania zagadki.

3. Projektowanie

3.1. Wykorzystane narzędzia

Podczas tworzenia aplikacji, której docelowym środowiskiem będzie Android wykorzystaliśmy platformę open source Xamarin i zestaw narzędzi Xamarin Community Toolkit, który ułatwił wykonywanie niektórych zadań. Projekt jest tworzony w języku C# co umożliwia nam wykorzystanie Microsoft Visual Studio Community Edition 2022. IDE (z ang. Integrated Development Environment), czyli zintegrowane środowisko programistyczne posłuży nam do łatwiejszego modułowania aplikacji. Wszystkie wersje kodów aplikacji znajdują się na platformie GitHub dzięki czemu łatwiej będzie wrócić do poprzednich wersji aplikacji w przypadku wystąpienia błędów w działaniu aktualnej. Wszystko początkowo miało zostać połączone za pomocą bazy danych Firebase. Baza ta umożliwiałaby przesyłanie danych między graczami i ułatwiała przejście do następnych poziomów.

3.1.1. Xamarin

Xamarin¹ (logo - rys. 3.1) to platforma typu open source do tworzenia nowoczesnych i wydajnych aplikacji dla systemów iOS, Android i Windows za pomocą platformy .NET. Xamarin to warstwa abstrakcji, która zarządza komunikacją udostępnionego kodu z bazowym kodem platformy. Środowisko Xamarin działa w środowisku zarządzanym, które zapewnia wygody, takie jak alokacja pamięci i odzyskiwanie pamięci.



Rys. 3.1. Xamarin

3.1.2. Xamarin Community Toolkit

Zestaw narzędzi Xamarin Community Toolkit² (logo - rys 3.2) to zbiór elementów wielokrotnego użytku na potrzeby tworzenia aplikacji mobilnych za pomocą zestawu

¹[http://xamarinlab.pl/\[www1\]](http://xamarinlab.pl/[www1])

²[https://learn.microsoft.com/pl-pl/xamarin/community-toolkit/\[www2\]](https://learn.microsoft.com/pl-pl/xamarin/community-toolkit/[www2])

narzędzi Xamarin.Forms, w tym animacji, zachowań, konwerterów, efektów i pomocników. Upraszcza i demonstrowuje typowe zadania deweloperskie podczas kompilowania aplikacji dla systemów iOS, Android, macOS, WPF i platforma uniwersalna systemu Windows (UWP) przy użyciu platformy Xamarin.Forms.



Rys. 3.2. Xamarin Community ToolKit

3.1.3. Microsoft Visual Studio

Microsoft Visual Studio³ (logo - rys 3.3) to zintegrowane środowisko programistyczne, służące do tworzenia, edytowania i debugowania kodu. IDE (z ang. Integrated Development Environment), czyli zintegrowane środowisko programistyczne to software oferujący szereg funkcji przydatnych podczas tworzenia oprogramowania dla systemów Windows, Android, iOS, rozwiązań internetowych oraz opartych o chmurę. Poza standardowym edytorem oraz debugerem, które zapewnia większość aplikacji IDE, program Microsoft Visual Studio zawiera jeszcze kompilatory, narzędzia do uzupełniania kodu i wiele innych funkcji usprawniających pracę.

³<https://visualstudio.microsoft.com/pl/>[www3]



Rys. 3.3. Visual

3.1.4. Git, GitHub

Git jak i GitHub⁴ (logo - rys 3.4) to najczęściej używany nowoczesny system kontroli wersji. Za pomocą usługi Git możesz śledzić zmiany kodu wprowadzane w czasie i przywrócić określone wersje. Ponadto pozwala kontrolować dostęp do danych, wspiera zarządzanie wieloma repozytoriami. Jest to rozwiązanie, które pozwala zaoszczędzić sporo czasu i nerwów. Kolejnym plusem jest to, że jest on w miarę łatwy w użyciu i jest przejrzysty.

⁴<https://github.com/>[www4]



Rys. 3.4. GitHub

3.1.5. Język C#

Język C#⁵ (logo - rys 3.5) to zorientowany obiektowo język programowania, który umożliwia programistom tworzenie różnych bezpiecznych i niezawodnych aplikacji w środowisku .NET. Język ten udostępnia konstrukcje językowe, które bezpośrednio obsługują te koncepcje, dzięki czemu język C# jest językiem naturalnym, w którym można tworzyć składniki oprogramowania i ich używać.

⁵<https://learn.microsoft.com/pl-pl/dotnet/csharp/>[www5]



Rys. 3.5. C#

3.1.6. Baza danych SQLite

SQLite⁶ (logo - rys 3.6) jest bez serwerową, relacyjną, lekką bazą danych. Znajduje się w dokładnie jednym pliku. Bardzo często wybierana jako baza dla aplikacji iOS oraz Android. Zawartość bazy danych przetrzymywana jest w jednym pliku. Baza SQLite jest utrzymywana na dysku przy użyciu B-drzew. Osobne drzewo jest używane dla każdej z tabel i każdego z indeksów. Baza udostępnia transakcje ACID oraz implementuje większość standardu SQL 92. Jest często wykorzystywany w większych aplikacjach oraz w systemach obsługi relacyjnych baz danych takich jak Kexi.



Rys. 3.6. Komunikat błędu

⁶<https://www.sqlite.org/index.html>/[www6]

3.2. Działanie aplikacji

Gra zawiera kilka paneli:

- Menu główne
- Menu ustawień
- Dwa tryby gry
 - Tryb graficzny
 - Tryb tekstowy

3.2.1. Menu główne

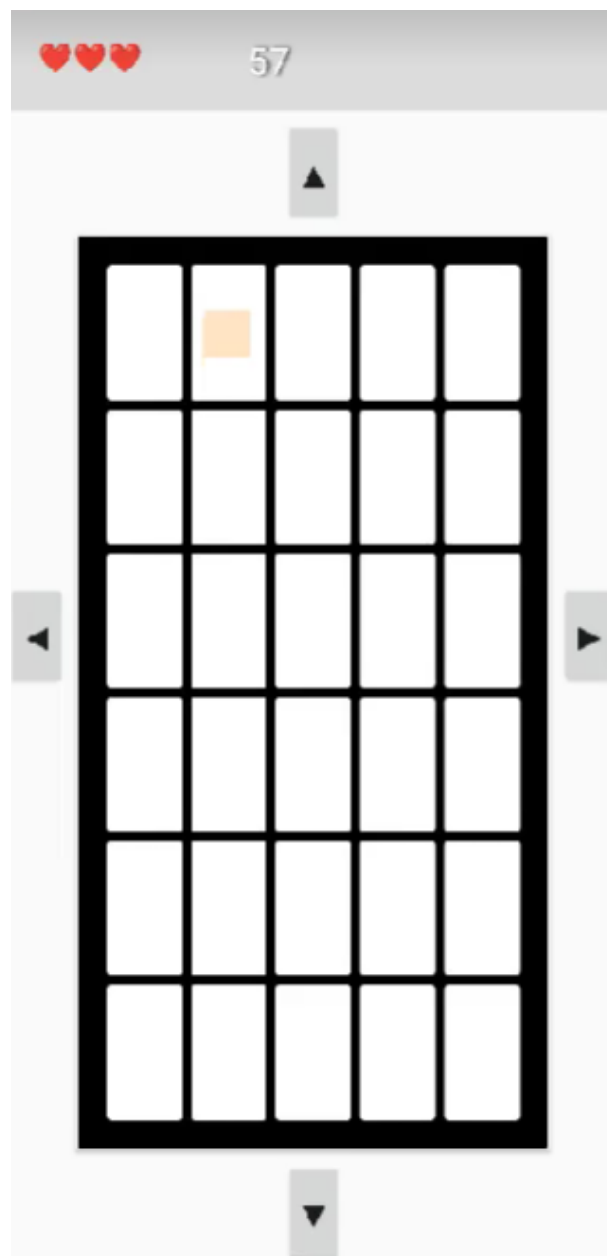
Po uruchomieniu aplikacji powita nas już pierwsza zagadka (rys 3.7), która polega na znalezieniu przycisku pozwalającego przejść dalej. Tym przyciskiem jest przycisk oznaczony żarówką. Jeśli jednak zostanie wciśnięty duży czerwony przycisk na środku ekranu to gra zostanie wyłączona.

Jak widać mamy dostępny jeszcze przycisk oznaczony zębatką. Prowadzi on do ustawień rozgrywki.



Rys. 3.7. Menu Główne

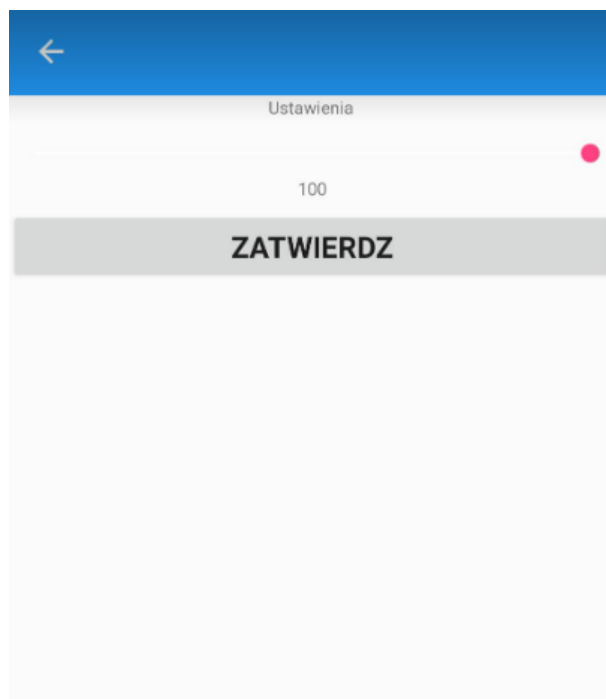
Po wciśnięciu dobrego przycisku przechodzimy do właściwego menu głównego (rys 3.8), w którym możemy przejść dalej.



Rys. 3.8. Menu Główne v2

3.2.2. Ustawienia

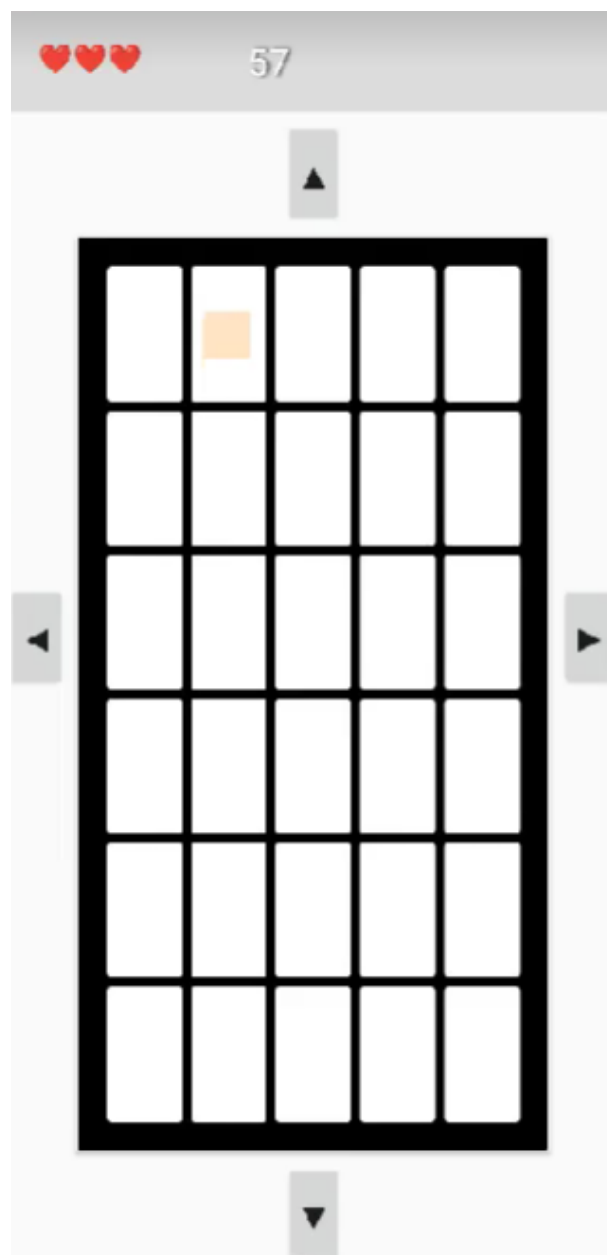
Ten panel (rys. 3.9) oferuje nam zmianę ustawień gry. Dla przykładu głośności muzyki. Po zmienieniu głośności w panelu ustawień wartość ustawiona zostaje zapisana i przenoszona na inne panele aplikacji. Przejście do menu ustawień będzie możliwe zarówno z menu głównego jak i z poziomu gry.



Rys. 3.9. Ustawienia

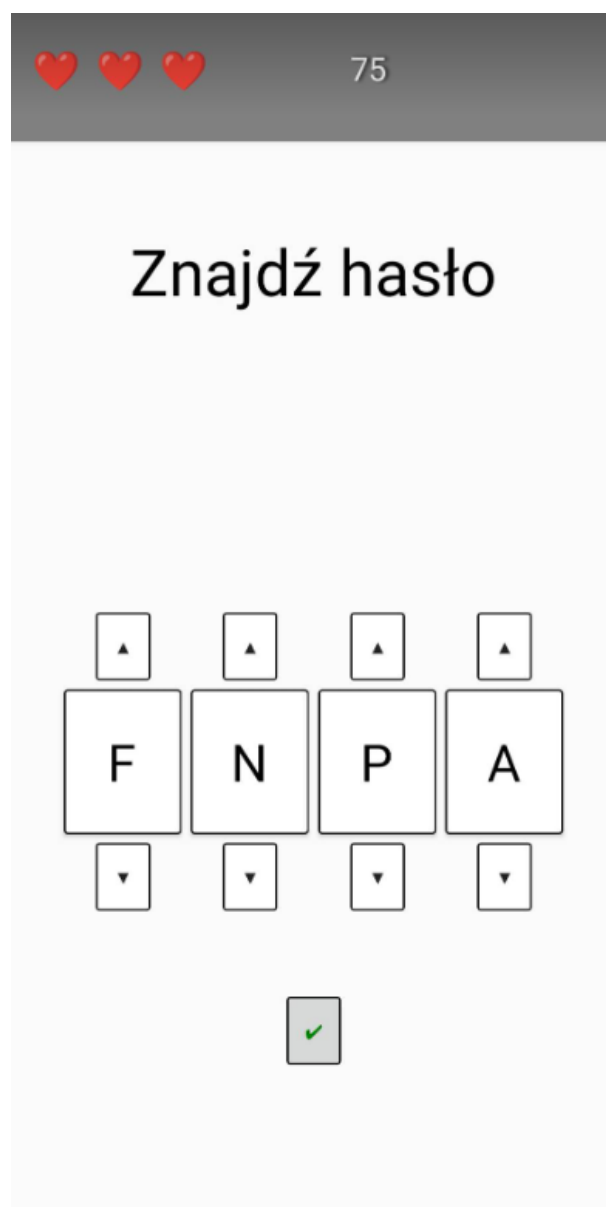
3.2.3. Tryb graficzny

Pierwsza zagadka w trybie graficznym będzie wyglądała tak jak na rys 3.10. Pomarańczowy kwadrat jest naszą postacią, którą poruszamy za pomocą strzałek umieszczonych na ekranie. Każda strzałka odpowiada za ruch w inną stronę.



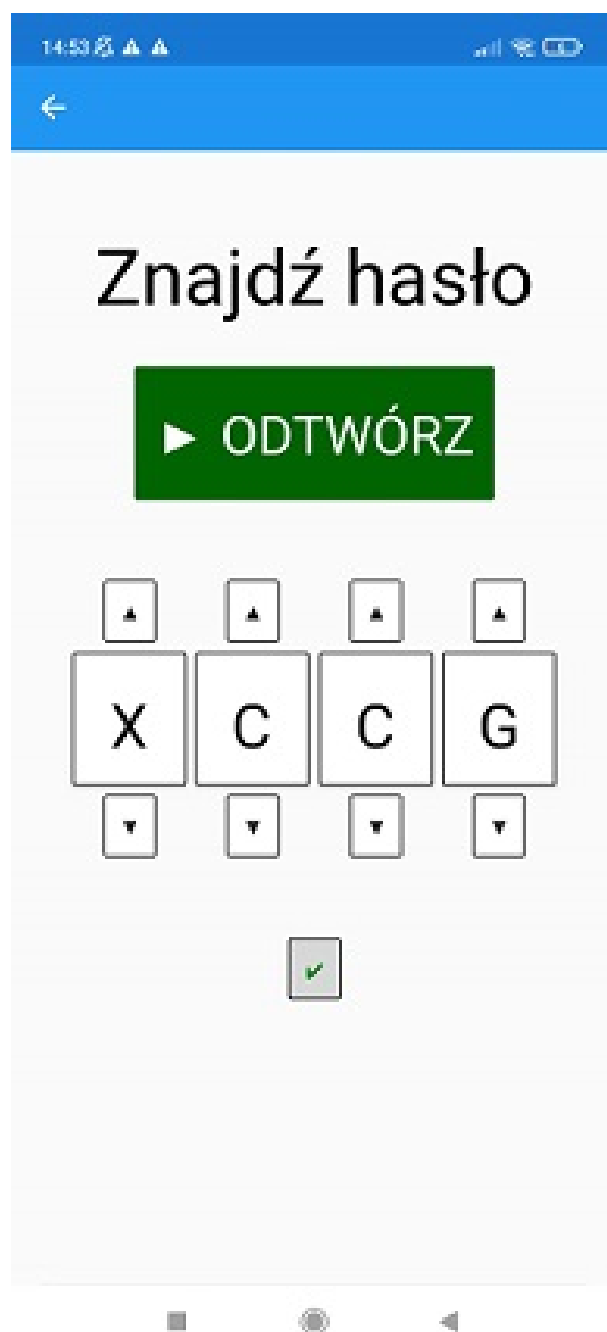
Rys. 3.10. Labirynt

Drugą zagadką będą "Literaki" (rys 3.11). Zagadka ta będzie polegała na ułożeniu odpowiedniego słowa. Za pomocą strzałek będziemy mogli zamieniać litery na odpowiednich polach. Przycisk z "ptaszkiem" pozwoli nam sprawdzić ułożone słowo.



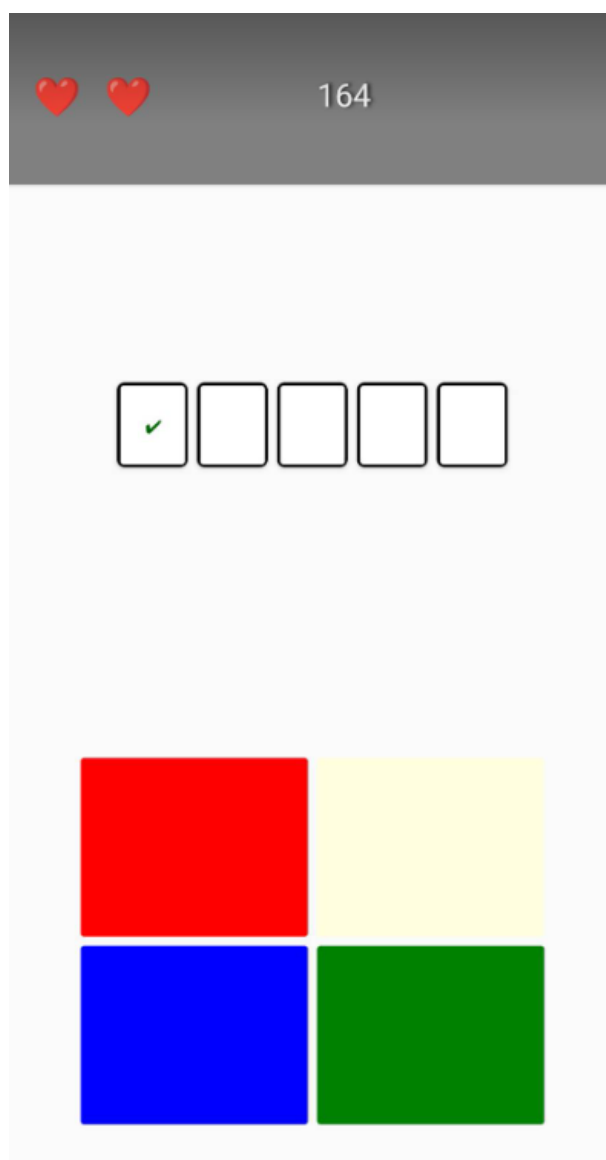
Rys. 3.11. Literaki

Kolejna zagadka będzie polegała na wyborze odpowiedniego słowa, będziemy zamieniać litery za pomocą strzałek jak na rys 3.12. Jeżeli ułożymy odpowiednie słowo zatwierdzamy przyciskiem oznaczonym "ptaszkiem". Za pomocą przycisku odtwórz możemy odtworzyć kod morsa z latarki od nowa.



Rys. 3.12. Kod morsa

Następna zagadka (rys 3.13) będzie polegała na wypatrywaniu jaki kolor został wyróżniony po czym kliknięcie w odpowiadający mu inny. Zagadka w zamyśle jest porosta ale testu pamięć i spostrzegawczość osoby grającej w trybie graficznym.



Rys. 3.13. Kolorki

Ostatnia zagadka będzie najprostsza (rys 3.14). Będzie polegała na spostrzeżeniu koloru przycisku i przytrzymaniu go przez określony czas.



Rys. 3.14. Wielki Przycisk

4. Implementacja

4.1. Przycisk

Przyciski to podstawowy i najprostszy element większości aplikacji. Ten element poprzez klikanie w niego wykonuje określoną akcję. Kod przykładowego przycisku w naszej grze można znaleźć na listunku 1:

```
1 <Button Grid.Row="0"  
2 Grid.Column="0"  
3 Text="&#x2699;"  
4 x:Name="SettingsButton"  
5 Padding="0,0,2,2"
```

```

6 FontAttributes="Bold"
7 FontSize="24"
8 Clicked="GoToSettings"></Button>

```

Listing 1. Button

4.1.1. Użyte atrybuty przycisku (listing 1)

- **Grid** - ustawia element na "siatce",
 - **Grid.Row** (linia 1) - odpowiada za oś Y czyli rząd,
 - **Grid.Column** (linia 2) - odpowiada za oś X czyli kolumnę,
- **Text** (linia 3) - napis wewnątrz przycisku. W tym wypadku kod szesnastkowy (hexcode) zębatki,
- **Padding** (linia 5) - wewnętrzny margines,
- **FontSize** (linia 7) - rozmiar czcionki,
- **Clicked** (linia 8) - przycisk po kliknięciu odwołuje się do metody o tej nazwie,
- **CornerRadius** - zaokrągla rogi przycisku, przy odpowiednio dużej wartości, przycisk może stać się okrągły.

4.1.2. Inne użyteczne atrybuty

Można znaleźć na listingu 2. Opisuja one polecenie label.

- **X:Name** (linia 3) - nazwa elementu używana potem w C#,
- **Vertical/HorizontalTextAlignment** (linia 6) - ustawia test na osi X/Y w zależności od wybranej opcji,
- **BackgroundColor** (linia 8) - pozwala na zmianę koloru tła.

```

1 <Label Grid.Row="0"
2   Grid.Column="0"
3   x:Name="Tit"
4   Text="Tytuł"
5   Grid.ColumnSpan="4"
6   VerticalTextAlignment="Center"
7   HorizontalTextAlignment="Center"
8   BackgroundColor="White"/>

```

Listing 2. Label**4.1.3. Przyciski w formie zdjęcia**

Inna forma wstawiania przycisku, w którym zamiast tekstu znajdziemy wstawione zdjęcie jest pokazana na listingu 3.

```

1 <ImageButton Grid.Column="0" Grid.Row="0" x:Name="BulbButton"
   Clicked="ChangeLight" ClassId="0"
2 Source="https://cdn-icons-png.flaticon.com/512/74/74072.png" Aspect
   ="AspectFit" IsVisible="true" >
3 <VisualStateManager.VisualStateGroups>
4   <VisualStateGroup x:Name="CommonStates">
5     <VisualState x:Name="Normal">
6       <VisualState.Setters>
7         <Setter Property="Scale" Value="1" />
8       </VisualState.Setters>
9     </VisualState>
10    <VisualState x:Name="Pressed">
11      <VisualState.Setters>
12        <Setter Property="Scale" Value="0.8" />
13      </VisualState.Setters>
14    </VisualState>
15  </VisualStateGroup>
16 </VisualStateManager.VisualStateGroups>
17 </ImageButton>

```

Listing 3. ImageButton**4.1.4. Użyte atrybuty**

- **VisualState (linie 3-16)** - pozwala na zmianę wyglądu w zależności od stanu przycisku. W naszym przypadku state "normal" odnosi się do domyślnego ustawienia. State "Pressed" sprawia, że skala obiektu zmienia się do 0.8 (zmniejsza się, linia 12) symulując efekt naciśnięcia po czym wraca do state "normal",
- **Source (linia 2)** - źródło obrazka, w tym wypadku link ale możliwe jest też użycie ścieżki.

4.2. MainPage

Funkcja pokazana na listingu 4 służy do sprawdzenia czy jest ustalona wartość głośności muzyki, a następnie ustawia w linii 6 wartość Volume soundtracku właśnie jako tą wartość.

```
1 public MainPage(double dalej)
2 {
3     InitializeComponent();
4     if (dalej == 0)
5         dalej = 1;
6     soundtrack.Volume = dalej;
7 }
```

Listing 4. MainPage

4.3. OnAppearing

Na listingu 5 nadpisujemy metodę OnAppearing(), która wykonuje się na starcie. Metoda FindByName znajduje elementy o podanych nazwach, na następnie wykonuje dla nich metodę Animate.Pulse().

Do zmiennej status przypisany zostaje stan "Uprawnienia do używania aparatu" przez aplikację (potrzebne do użycia lampy błyskowej).

Następnie w linii 16 sprawdzany jest status. Jeżeli status jest nieznany (unknown) lub odmówiono dostępu aplikacja prosi o przyznanie takiego uprawnienia.

```
1 async protected override void OnAppearing()
2 {
3     base.OnAppearing();
4     arrow_1 = (Label)FindByName("arrow_1");
5     arrow_2 = (Label)FindByName("arrow_2");
6     arrow_3 = (Label)FindByName("arrow_3");
7     arrow_4 = (Label)FindByName("arrow_4");
8     bigRed = (Button)FindByName("bigRed");
9     BulbButton = (ImageButton)FindByName("BulbButton");
10    Animate_pulse(arrow_1);
11    Animate_pulse(arrow_2);
12    Animate_pulse(arrow_3);
13    Animate_pulse(arrow_4);
14    animate_button(bigRed);
15    var status= await Permissions.CheckStatusAsync<Permissions.Camera>();
16    if (status==PermissionStatus.Unknown || status ==
        PermissionStatus.Denied)
```

```

17 {
18     await Permissions.RequestAsync<Permissions.Camera>();
19 }
20 }

```

Listing 5. OnAppearing

4.4. Animate_pulse

Animate_pulse zawarte na listingu 6 włącza animację, w której to skala strzałek zmienia się pomiędzy 1 a 1.2.

W tym przypadku używane jest ClassId po to, by nazywać animacje różnymi nazwami (te same nazwy nadpisywałyby się).

W tym przypadku, przez połowę czasu animacji skala rośnie, by przez drugą połowę maleć.

- **This (linia 9)** – Animacja, która będzie animowana,
- **Objname (linia 10)** – nazwa służąca jako uchwyt do animacji,
- **16 (linia 11)** – Czas w milisekundach pomiędzy elementami animacji,
- **950 (linia 12)** – czas w milisekundach trwania animacji,
- **Easing.Linear (linia 13)** – funkcja używana do opisu animacji,
- **(v,c) =>obj.Scale=1 (linia 14)** – co ma zostać wykonane po skończeniu animacji ,
- **()=>true (linia 15)** – czy funkcja ma być zapętłona.

```

1 void Animate_pulse(Label obj)
2 {
3     var Objname = obj.ClassId;
4     var a = new Animation {
5         {0,0.5, new Animation(v => obj.Scale = v, 1, 1.2) },
6         {0.5,1, new Animation(v => obj.Scale = v, 1.2, 1) }
7     };
8     a.Commit(
9         this,
10        Objname,
11        16,
12        950,
13        Easing.Linear,

```



```

14 (v, c) => obj.Scale = 1,
15 ()=> true);
16 }

```

Listing 6. Animate_pulse

4.5. FirstTask

Funkcja zawarta na listingu 7 to funkcja, przyjmująca jako parametry obiekt, który został wciśnięty i argumenty zdarzenia, pozwala na przemieszczanie się pomiędzy stronami aplikacji.

Metoda soundtrack.Stop() (linia 3) sprawia, że soundtrack przestaje grać.

Do zmiennej dalej (linia 4) przekazywana jest aktualna wartość głośności muzyki granej w tle.

Następnie (linia 5) asynchronicznie przechodzimy do nowej strony, przekazując jako parametr głośność soundtracku, by ten mógł grać z taką samą głośnością jaka była ustawiona.

```

1 private async void FirstTask(object sender, EventArgs e)
2 {
3     soundtrack.Stop();
4     double dalej1 = soundtrack.Volume;
5     await Navigation.PushAsync(new LabiryntPage(dalej1));
6 }

```

Listing 7. FirstTask

4.6. Wciśnięcie złego przycisku

Funkcja na listingu 8 odpowiada za wciśnięciu złego przycisku. Po wciśnięciu go zmienia się widoczność wszystkich elementów menu, następnie wyświetla się obrazek świadczący o porażce (linia 17), a po odczekaniu 3,5s (linia 19) proces aplikacji kończy się i zwraca wartość 0, co świadczy o poprawnym zamknięciu aplikacji (linia 20).

```

1 async void WrongButtonClicked(object sender, EventArgs e)
2 {
3     ChangeLight(sender,e);
4     var ButtonDalej = (Button)FindByName("LabiryntButton");
5     var bulbButton = (ImageButton)FindByName("BulbButton");
6     var settingsButton= (Button)FindByName("SettingsButton");
7     var deadAnimation = (Image)FindByName("deadAnimation");
8     var ButtonLatarka = (Button)FindByName("FlashlightButton");

```

```

9  var ButtonTekstowy = (Button)FindByName("TextModeButton");
10 var ButtonMorse = (Button)FindByName("ButtonMorse");
11 ButtonDalej.IsVisible = false;
12 ButtonTekstowy.IsVisible = false;
13 ButtonLatarka.IsVisible = false;
14 ButtonMorse.IsVisible = false;
15 bulbButton.IsVisible = false;
16 settingsButton.IsVisible = false;
17 deadAnimation.IsVisible = true;
18 deadAnimation.IsAnimationPlaying = true;
19 await Task.Delay(3500);
20 System.Environment.Exit(0);
21 }

```

Listing 8. WrongButton

4.7. HUD

HUD (ang. heads-up display) – Ważna część gier komputerowych. Jest to sekcja zawierająca informacje na temat rozgrywki, pozwalająca graczowi na odnalezienie informacji o niej. Nazwa wywodzi się od tzw. wyświetlacza przeziernego HUD stosowanego np. przez pilotów samolotów bojowych.

W linach 5-7 są opisane życia w postaci serc.

Linie 8-9 są odpowiedzialne za timer, jego widoczność, przezroczystość i cień jaki rzuca.

```

1      <Frame Grid.Row="0" Grid.ColumnSpan="20" Grid.RowSpan
      = "2" BackgroundColor="Gray">
2  <Grid>
3  <Label Grid.Column="0"></Label>
4
5  <Label Grid.Column="0" TextColor="White" FontSize="30"
      VerticalTextAlignment="Center" x:Name="HealthBar1" Text
      = "&#10084;"></Label>
6  <Label Grid.Column="1" TextColor="White" FontSize="30"
      VerticalTextAlignment="Center" x:Name="HealthBar2" Text
      = "&#10084;"></Label>
7  <Label Grid.Column="2" TextColor="White" FontSize="30"
      VerticalTextAlignment="Center" x:Name="HealthBar3" Text
      = "&#10084;"></Label>
8
9  <Label x:Name="Timer" Margin="0,0,0,15" Grid.Column="3" Grid.
      ColumnSpan="3" TextColor="White" VerticalTextAlignment="Center"
      xct:ShadowEffect.Color="Black" xct:ShadowEffect.Opacity="0.7"

```

```

    xct:ShadowEffect.OffsetX="5" xct:ShadowEffect.Radius="5"
    FontSize="25" FontAttributes="Bold"></Label>
10 <Label x:Name="TimerCount" xct:ShadowEffect.Color="Black" xct:
    ShadowEffect.Opacity="0.7" xct:ShadowEffect.OffsetX="5" xct:
    ShadowEffect.Radius="5" Margin="0" TextColor="White" Grid.Column
    ="5" VerticalOptions="Center" FontSize="25"></Label>
11 <Label Grid.Column="5" Grid.ColumnSpan="3" TextColor="White"
    VerticalTextAlignment="Center" x:Name="CodeNumber"></Label>
12
13 <Label Grid.Column="1"></Label>
14 </Grid>
15 </Frame>

```

Listing 9. HUD

4.8. Literaki

Linia 1 - zmienna zawierająca ilość "życ" (podejść),

Linia 2 - tablica przechowująca indeks każdego z bloków,

Linia 3 - tablica przechowująca słowa mogące wystąpić jako rozwiązanie,

Linia 4 - tablica przechowująca wszystkie wartości indeksów każdego z bloków,

Linia 11 - zmienna zawierająca indeks wartości z TextTable[], która została wybrana jako rozwiązanie,

Linia 12 - zmienna zawierająca pozostały czas na rozwiązanie zagadki,

Linia 13 - zmienna zawierająca obiekt klasy Random, służący m.in. do losowania liczby z przedziału.

```

1 public int HealthCount = 3;
2 int[] BlockCounters = { 0, 0, 0, 0 };
3 readonly string[] TextTable = { "ARAB", "ALGA", "ALFA", "BAZA", "
    BETA", "BUDA", "BUNT", "GLON", "GONG", "GRAM", "LAWA", "LIRA", "
    LUFA", "RATA", "ROPA", "RYSY" };
4 public char[,] WordTable =
5 {
6     {'A','A','A','A'},
7     {'A','A','A','A'},
8     {'A','A','A','A'},
9     {'A','A','A','A'},
10 };
11 public int ChosenWord;
12 public int TimeLeft = 60;
13 Random RandomCharCount = new Random();

```

Listing 10. Literaki

4.9. SetTime

W Linii 4 wyświetla się okno, które należy zamknąć, zanim ruszy zegar. Później uruchamiana jest metoda `StartTimer(TimeSpan)`. `TimeSpan` przyjmuje trzy wartości - godziny, minuty i sekundy.

W Liniach 8-9 Odejmowane jest 1 od pozostałego czasu, a następnie uzyskana wartość jest wpisywana jako `text` do obiektu `TimeCount`.

W linii 10 widoczny jest warunek, który zmienia kolor tekstu na czerwony jeśli mamy mniej czasu niż 15 sekund, a w wypadku gdy się skończy czas wyświetla odpowiedni komunikat.

```

1 async void SetTime()
2 {
3     await DisplayAlert("Rozpocznij zagadke", "Wcisnij OK, aby
        rozpoczac", "OK");
4     Label TimeCount = (Label)FindByName("TimerCount");
5     Label TimeBar = (Label)FindByName("Timer");
6     Device.StartTimer(new TimeSpan(0, 0, 1), () =>
7     {
8         TimeLeft--;
9         TimeCount.Text = TimeLeft.ToString();
10        if (TimeLeft < 15)
11        {
12            TimeCount.TextColor = Color.Red;
13            if (TimeLeft == 0)
14            {
15                DisplayAlert("Przegrales", "):", "No nie");
16                TimeCount.Text = "";
17                TimeBar.Text = TimeCount.Text;
18                return false;
19            }
20        }
21        return true;
22    });
23 }
```

Listing 11. SetTime

4.10. SetColumns

W każdej iteracji pętli do zmiennej `name` wpisywana jest nazwa Bloku, po czym w zmiennej `Obj` umieszczany jest obiekt o tej nazwie.

W Linii 6 Losowany jest liczba stałoprzecinkowa z zakresu 0 do rozmiaru tablicy

WordTable.

W linii 7 jako wartość atrybutu text umieszczamy wylosowany znak.

Ta funkcja służy początkowemu ustawieniu tekstu w blokach.

```
1 setColumns()
2 {
3     for (int i = 0; i < 4; i++)
4     {
5         var Name = "Block" + i;
6         var Obj = (Label)FindByName(Name);
7         var Los = RandomCharCount.Next(0, (WordTable.Length / 4) - 1);
8         Obj.Text = WordTable[Los, i].ToString();
9     }
10 }
```

Listing 12. SetColumns

4.11. RandomizeWordTable

W Linii 7 do zmiennej nextChar wpisywana jest losowa liczba z zakresu (65,90), która następnie zostaje przekonwertowana na char (są to znaki z zakresu [A-Z]) i wpisana jako wartość tablicy WordTable - w ten sposób za każdym razem losowana jest cała tablica ze znakami.

```
1 RandomizeWordTable()
2 {
3     for (int i = 0; i < (WordTable.Length/4); i++)
4     {
5         for (int j = 0; j < 4; j++)
6         {
7             var nextChar = RandomCharCount.Next(65, 90);
8             WordTable[i, j] = Convert.ToChar(nextChar);
9         }
10    }
11    char[] chosenWordChars = TextTable[ChosenWord].ToCharArray();
12    for (int i = 0; i < 4; i++)
13    {
14        int random = RandomCharCount.Next(1, (TextTable.Length/4) - 1);
15        WordTable[random, i] = Convert.ToChar(chosenWordChars[i]);
16    }
17 }
```

Listing 13. RandomizeWordTable

4.12. RandomizeText

Sprawia, że do zmiennej ChosenWord wpisywana jest losowa liczba z zakresu od zera do długości tablicy ze słowami. Dzięki temu wylosowane zostało jedno słowo, które zostanie potem użyte jako hasło. RandomizeWordTable().

```
1 RandomizeWordTable()
2 {
3     for (int i = 0; i < (WordTable.Length/4); i++)
4     {
5         for (int j = 0; j < 4; j++)
6         {
7             var nextChar = RandomCharCount.Next(65, 90);
8             WordTable[i, j] = Convert.ToChar(nextChar);
9         }
10    }
11    char[] chosenWordChars = TextTable[ChosenWord].ToCharArray();
12    for (int i = 0; i < 4; i++)
13    {
14        int random = RandomCharCount.Next(1, (TextTable.Length/4) -
15        1);
16        WordTable[random, i] = Convert.ToChar(chosenWordChars[i]);
17    }
18 }
```

Listing 14. RandomizeText

4.13. MoveUp

Ta funkcja zbiera informacje na temat bloku o danym ClassId (każdy blok ma swój classId 0-3, classId jest takie same dla przycisku i bloku) i wysyła je do funkcji ChangeBlockText() z parametrem up.

```
1 void MoveUp(object sender, EventArgs e)
2 {
3     var Button = (Button)sender;
4     var Name = "Block" + Button.ClassId.ToString();
5     var whichBlock = (Label)FindByName(Name);
6     int ClassId = Convert.ToInt32(Button.ClassId);
7     ChangeBlockText(whichBlock, ClassId, "up");
8 }
```

Listing 15. MoveUp

4.14. ChangeBlockText

W zależności od podanego parametru, wartość indeksu `BlockCounters[blockNumber]` zwiększa się (dla parametru `down`) lub zmniejsza się (dla parametru `up`) sprawiając efekt "przewijania się" po liście znaków.

Jeżeli będziemy chcieli wyjść poza zakres znaków w tablicy, to wartość indeksu zmieni się na pierwszy (dla parametru `dow`) lub ostatni (dla parametru `up`).

```

1      void ChangeBlockText(Label block, int blockNumber, string
2      where)
3      {
4          if (where == "up")
5          {
6              if (BlockCounters[blockNumber] == 0)
7              {
8                  BlockCounters[blockNumber] = (WordTable.Length / 4) - 1;
9              }
10             else BlockCounters[blockNumber]--;
11         }
12         else if (where == "down")
13         {
14             if (BlockCounters[blockNumber] == (WordTable.Length / 4) - 1)
15             {
16                 BlockCounters[blockNumber] = 0;
17             }
18             else BlockCounters[blockNumber]++;
19         }
20         block.Text = WordTable[BlockCounters[blockNumber], blockNumber
21         ].ToString();

```

Listing 16. ChangeBlockText

4.15. CheckIfWin

Ta funkcja sprawdza warunek zwycięstwa (ciąg znaków zgadza się z elementem z tablicy `WordTable`).

W linii 4 ustawiana jest flaga `WinFlag` jako `false` (`false` oznacza brak zgodności, `true` zgodność).

W liniach 5-10 odczytywana jest wartość z elementów `Block0` - `Block3`, po czym jest to sklejane w jeden ciąg, umieszczony w zmiennej `userGuess`.

W liniach 11-17 iterujemy po wszystkich elementach tablicy `WordTable` (tam są

słowa możliwe do uzyskania).

W linii 13 umieszczony jest warunek, który sprawdza, czy wybrany przez użytkownika ciąg znaków jest zgodny z elementem z tablicy. Jeśli tak to flaga WinFlag przestawiana jest na "true".

W linii 18 sprawdzamy czy flaga ma wartość True. Jeśli tak, oznacza to wygraną. W przeciwnym razie zmniejszana jest liczba serc o 1.

```

1  async void CheckIfWin(object sender, EventArgs e)
2  {
3      var userGuess = "";
4      var WinFlag = false;
5      for (int i = 0; i < 4; i++)
6      {
7          var Name = "Block" + i;
8          var Block = (Label)FindByName(Name);
9          userGuess += Block.Text;
10     }
11     for (int i = 0; i < TextTable.Length; i++)
12     {
13         if (TextTable[i] == userGuess)
14         {
15             WinFlag = true;
16         }
17     }
18     if (WinFlag)
19     {
20         DisplayAlert("Kod do następnej zagadki to: ", "2115", "Ok");
21         await Navigation.PushAsync(new FlashlightPage());
22     }
23     else
24     {
25         setHeartStatus();
26         LoverHeartCount();
27     }
28 }

```

Listing 17. CheckIfWin

4.16. MoveDown

Ta funkcja zbiera informacje na temat bloku o danym ClassId (każdy blok ma swój classId 0-3, classId jest takie same dla przycisku i bloku) i wysyła je do funkcji ChangeBlockText() z parametrem down.


```
1      void MoveDown(object sender, EventArgs e)
2  {
3      var Button = (Button)sender;
4      var Name = "Block" + Button.ClassId.ToString();
5      var whichBlock = (Label)FindByName(Name);
6      int ClassId = Convert.ToInt32(Button.ClassId);
7      ChangeBlockText(whichBlock, ClassId, "down");
8  }
```

Listing 18. MoveDown

5. Testowanie

Testy zostały wykonane według tabeli 5.1 i zostały w niej opisane

Tab. 5.1

Obiekt	Opis działania	Poprawność działania
Duży czerwony przycisk ("wciśnij mnie")	Wyłącza aplikacje	Działa poprawnie
Przycisk ustawień	Włącza menu ustawień	Działa poprawnie
Przycisk przejścia dalej (żarówka)	Włącza prawdziwe menu główne	Działa poprawnie
Przycisk "Przejdź dalej"	Przechodzi dalej	Działa poprawnie
Pasek zmiany głośności (Menu ustawień)	Podczas przesuwania zmienia wartość	Działa poprawnie
Przycisk zatwierdź (Menu ustawień)	Zapisuje wartość ustawioną na pasku i przypisuje ją dla każdej z kart	Działa poprawnie
Labirynt		
Strzałka w lewo (Labirynt)	Postać przemieszcza się w lewo	Działa poprawnie
Strzałka w prawo (Labirynt)	Postać przemieszcza się w prawo	Działa poprawnie
Strzałka w górę (Labirynt)	Postać przemieszcza się w górę	Działa poprawnie
Strzałka w dół (Labirynt)	Postać przemieszcza się w dół	Działa poprawnie
Przejście labiryntu (Wejście na odpowiednie pole)	Przenosi do innej zagadki	Działa poprawnie
Wejście w ścianę	Resetuje pozycję postaci i odbiera jedno z żyć	Działa poprawnie
Kod morsa		
Strzałka w górę (Pierwsza)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w górę (Druga)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w górę (Trzecia)	Zmienia literę na odpowiednim polu	Działa poprawnie

Strzałka w górę (Czwarta)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w dół (Pierwsza)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w dół (Druga)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w dół (Trzecia)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w dół (Czwarta)	Zmienia literę na odpowiednim polu	Działa poprawnie
Przycisk odtwórz	Odtwarza kod morsa za pomocą latarki	Działa poprawnie
Zatwierdzenie hasła	W przypadku podania błędного hasła zabiera życie natomiast w przypadku podania dobrego hasła przechodzi dalej	Działa poprawnie
Kolorki		
Lewy górny przycisk (czerwony)	W przypadku kliknięcia w błędnej kolejności zabiera życie natomiast w przypadku kliknięcia w dobrej kolejności daje punkt lub przechodzi dalej	Działa poprawnie
Prawy górny przycisk (żółty)	W przypadku kliknięcia w błędnej kolejności zabiera życie natomiast w przypadku kliknięcia w dobrej kolejności daje punkt lub przechodzi dalej	Działa poprawnie
Lewy dolny przycisk (niebieski)	W przypadku kliknięcia w błędnej kolejności zabiera życie natomiast w przypadku kliknięcia w dobrej kolejności daje punkt lub przechodzi dalej	Działa poprawnie

Prawy dolny przycisk (zielony)	W przypadku kliknięcia w błędnej kolejności zabiera życie natomiast w przypadku kliknięcia w dobrej kolejności daje punkt lub przechodzi dalej	Działa poprawnie
Zdobycie pięciu punktów	Przechodzi dalej	Działa poprawnie
Strata żyć	Przegranie gry	Działa poprawnie
Timer	Liczy czas	Działa poprawnie
Literaki		
Strzałka w górę (Pierwsza)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w górę (Druga)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w górę (Trzecia)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w górę (Czwarta)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w dół (Pierwsza)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w dół (Druga)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w dół (Trzecia)	Zmienia literę na odpowiednim polu	Działa poprawnie
Strzałka w dół (Czwarta)	Zmienia literę na odpowiednim polu	Działa poprawnie
Zatwierdzenie hasła	W przypadku podania błędnego hasła zabiera życie natomiast w przypadku podania dobrego hasła przechodzi dalej	Działa poprawnie
Duży zakres liter	Zakres liter jest taki sam na każdym polu	Działa poprawnie
Pozostałe		
Przycisk "Przejdź do menu"	Przechodzi do menu	Działa poprawnie
Wielki przycisk (Zagadka)	Poprawnie przytrzymany przechodzi do tabeli wyników	Działa poprawnie

Tabela wyników	Zawiera wyniki poprzednich graczy	Działa poprawnie
-------------------	--------------------------------------	------------------

6. Podręcznik użytkownika

6.1. Menu główne

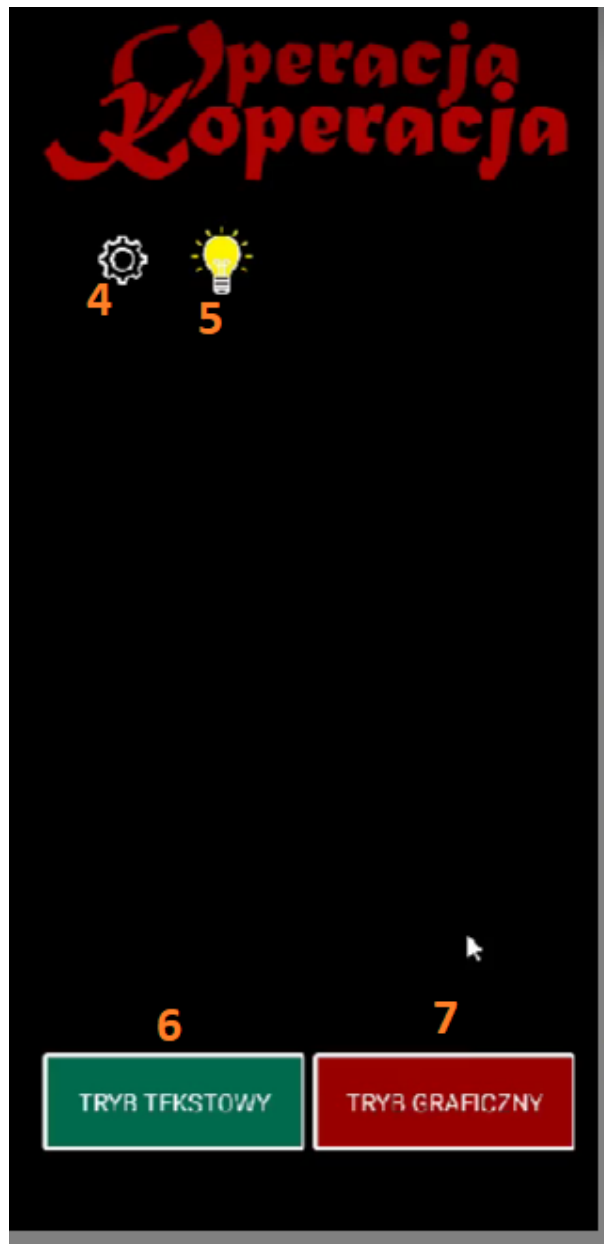
Menu główne w aplikacji "Operacja Kooperacja" zawiera pierwszą zagadkę w grze. Jest nią duży czerwony przycisk z napisem "CLICK HERE". Po wciśnięciu tego przycisku aplikacja zamyka się natomiast przycisk z "żarówką" pozwala nam przejść dalej.



Rys. 6.1. Menu Główne

1. Przycisk pozwalający nam przejść do ustawień,

2. Przycisk pozwalający nam przejść do właściwego menu głównego,
3. Przycisk ten wyłącza aplikację,

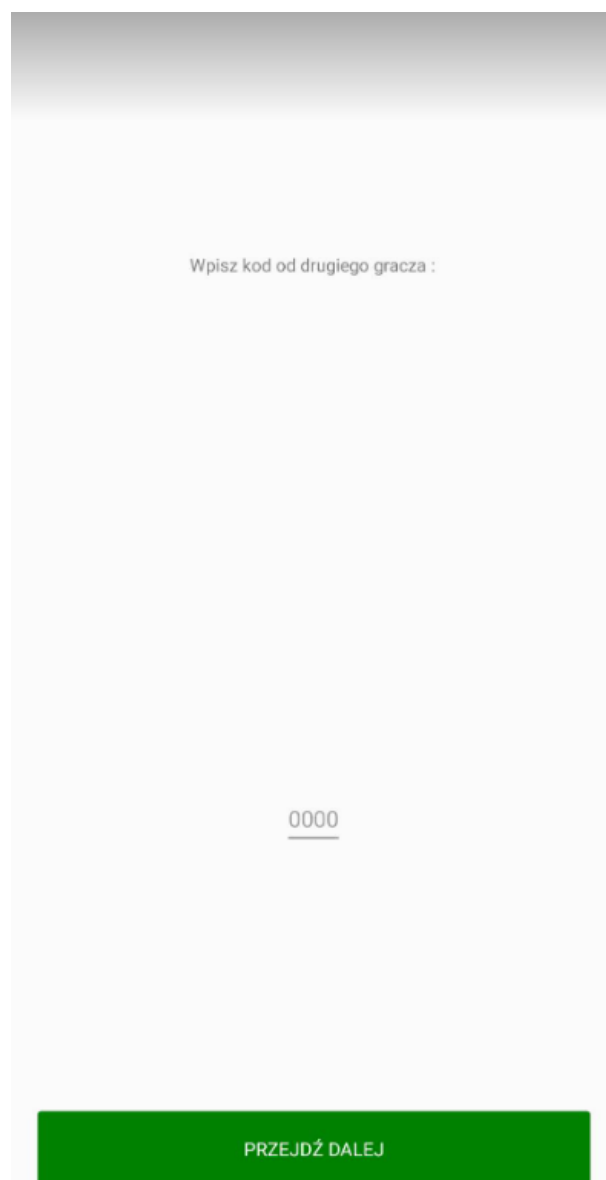


Rys. 6.2. Menu Główne v2

4. Przycisk pozwalający nam przejść do ustawień,
5. Przycisk pozwalający nam przejść do poprzedniego menu głównego,
6. Przycisk ten pozwala nam wybrać tryb tekstowy jako nasz tryb gry,
7. Przycisk ten pozwala nam wybrać tryb graficzny jako nasz tryb gry.

6.2. Tryb tekstowy

Po wejściu w tryb tekstowy ukazuje nam się takie okno jak na rys 6.3. Tutaj wprowadzamy kod w odpowiednim miejscu, który pozwoli nam zobaczyć odpowiedź do danej zagadki. Kod jest widoczny w trybie graficznym razem z opisem zagadki w komunikacie przed rozpoczęciem próby rozwiązania zagadki.

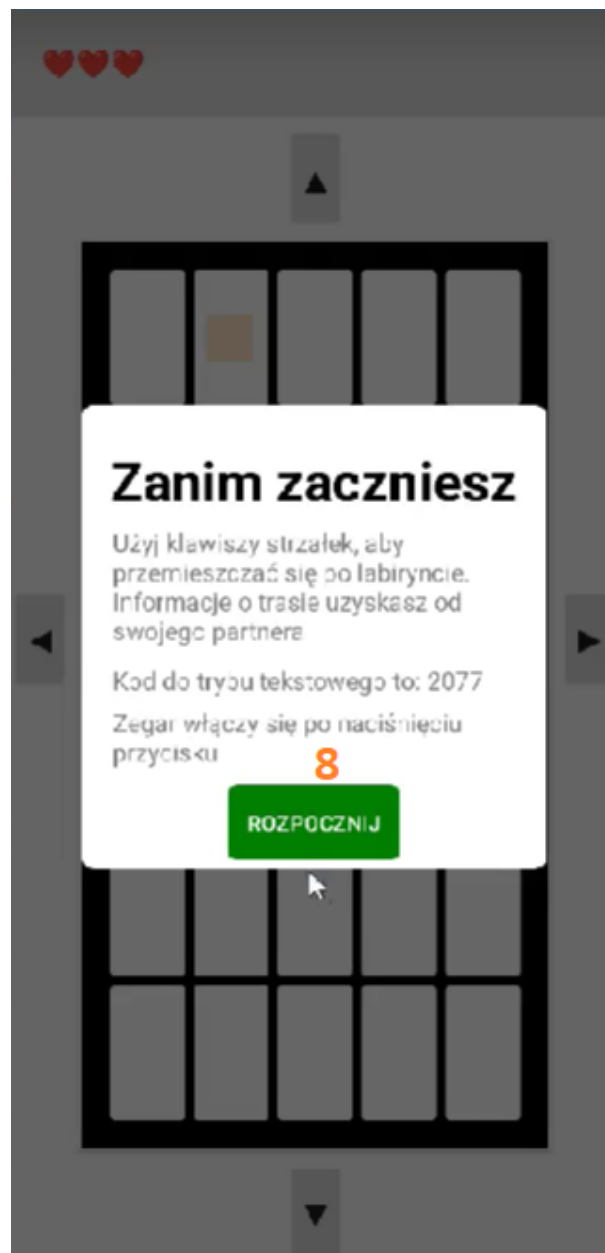


Rys. 6.3. Tryb tekstowy

6.3. Labirynt

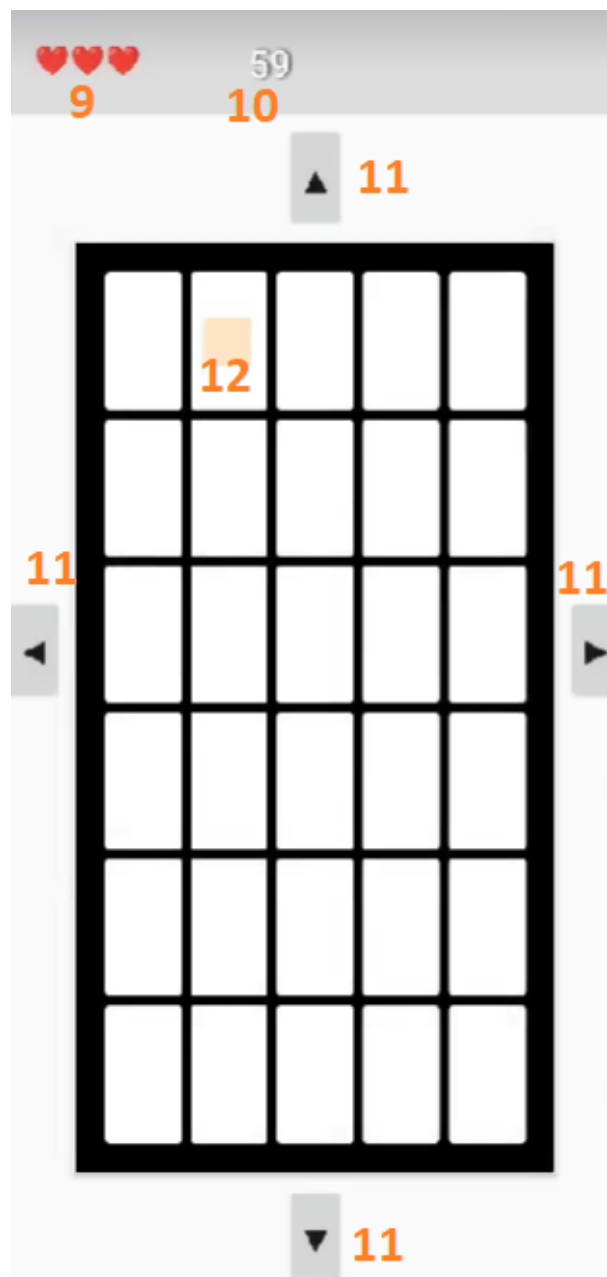
Po wybraniu graficznego trybu gry zostajemy przeniesieni do pierwszej zagadki, którą jest labirynt. Na początku dostajemy informacje o zagadce jak i kod do trybu

tekstowego dla naszego partnera. Ta zagadka polega na doprowadzeniu postaci (pomarańczowego kwadratu) do wyjścia. Osoba w trybie graficznym widzi kilka wersji labiryntu i musi wybrać właściwy na podstawie informacji jakie dostanie od partnera, po czym musi poprowadzić osobę z trybu graficznego do wyjścia. Każde wejście w ścianę kończy się utratą życia i zresetowaniem pozycji postaci.



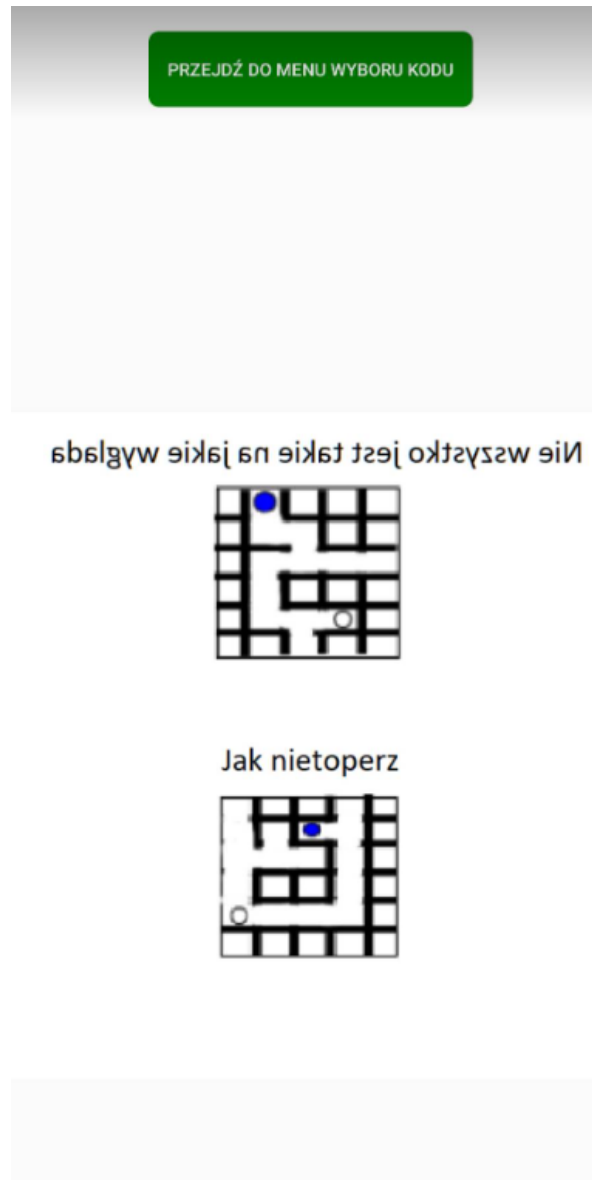
Rys. 6.4. Komunikat (Labirynt)

8. Przycisk pozwalający nam rozpocząć grę,



Rys. 6.5. Labirynt

9. Licznik żyć (Zaczynamy mając 3 życia, w momencie utraty ostatniego przegrywamy),
10. Timer (Pokazuje ile czasu mamy na daną zagadkę),
11. Przyciski pozwalające poruszać naszą "postać",
12. Postać, którą sterujemy.



Rys. 6.6. Tryb tekstowy (Labirynt)

Jak widać na rys 6.6 tryb tekstowy do zagadki "Labirynt" przedstawia możliwe rozłożenia labiryntu ale w tym trybie też są zagadki. Dla przykładu pierwsze rozłożenie labiryntu jest odbite lustrzanie od właściwego a drugie jest do góry nogami.

6.4. Literaki

Po przejściu "Labiryntu" zostajemy przeniesieni do kolejnej zagadki, którą są "Literaki". Podobnie jak w przypadku poprzedniej zagadki dostajemy opis tego co mamy zrobić i kod do trybu tekstowego. W tej zagadce chodzi o to, żeby razem z partnerem z trybu tekstowego ustalić jakie hasło pozwoli nam przejść dalej. Gracz w trybie tek-

sowym widzi dostępne słowa natomiast gracz w trybie graficznym musi za pomocą strzałek zmieniać litery na odpowiednich polach dopóki nie powstanie z nich jedno z tych słów. Liczba liter przypadająca na każde pole jest ograniczona.



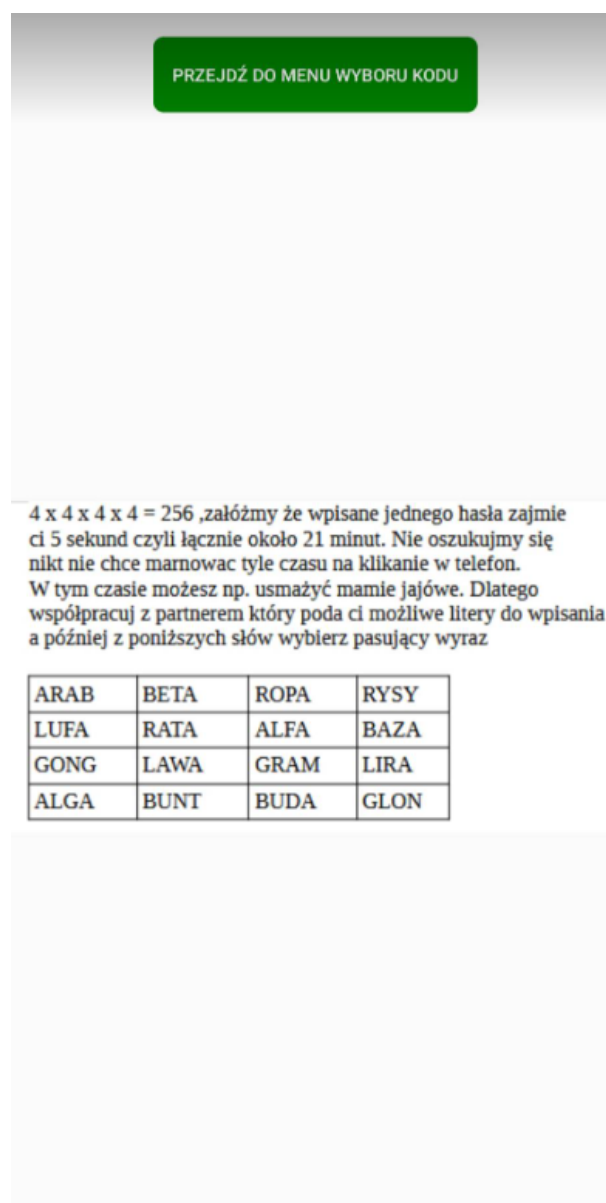
Rys. 6.7. Literaki

13. Przycisk, którym zatwierdzamy hasło,
14. Przyciski pozwalające na zmianę litery na odpowiednim miejscu.

Tryb tekstowy dla zagadki "Literaki" (rys 6.9) zawiera tabelę możliwych słów, autorski opis dlaczego nie warto próbować na siłę wpisywać hasła i dlaczego lepiej jest współpracować.



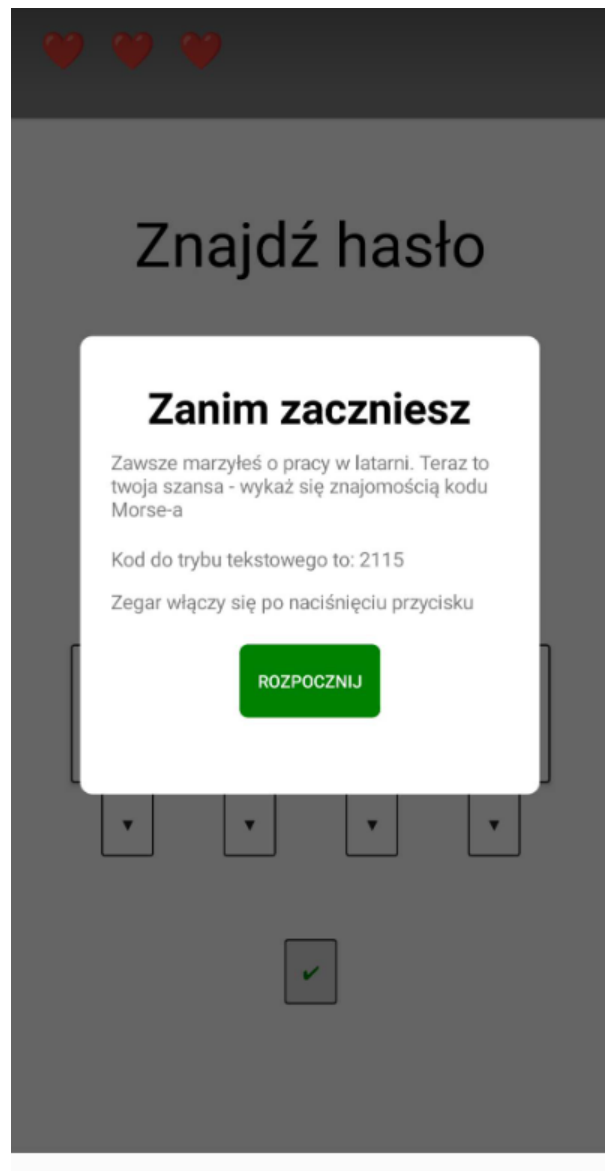
Rys. 6.8. Komunikat (Literaki))



Rys. 6.9. Tryb tekstowy (Literaki)

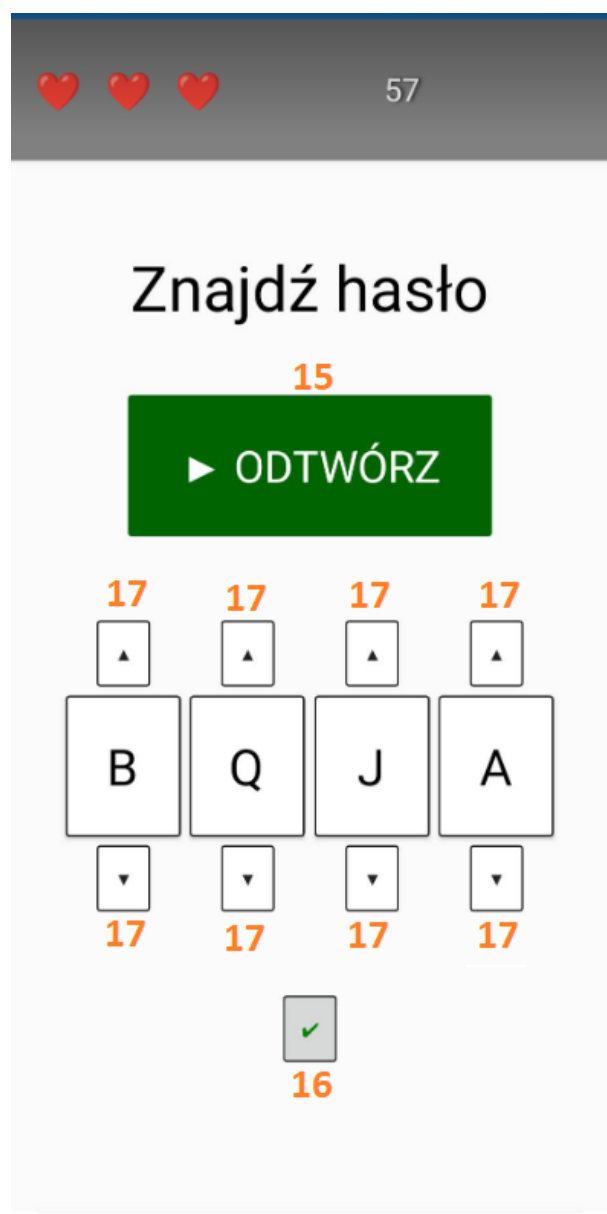
6.5. Kod Morsa

Po ukończeniu "Literaków" zostajemy przeniesieni do zagadki, która nazywa się "Kod Morsa". Podobnie jak w przypadku poprzednich zagadek dostajemy opis zadań i kod do trybu tekstowego (rys 6.10). Ta zagadka jest podobna do "Literaków" z jedną różnicą. Gracz w trybie tekstowym ma opisane słowa za pomocą kodu morsa. Gracz trybu graficznego dostaje, poprzez włączanie i wyłączanie latarki, długie i krótkie sygnały, które po skonsultowaniu z partnerem pozwolą odgadnąć hasło.



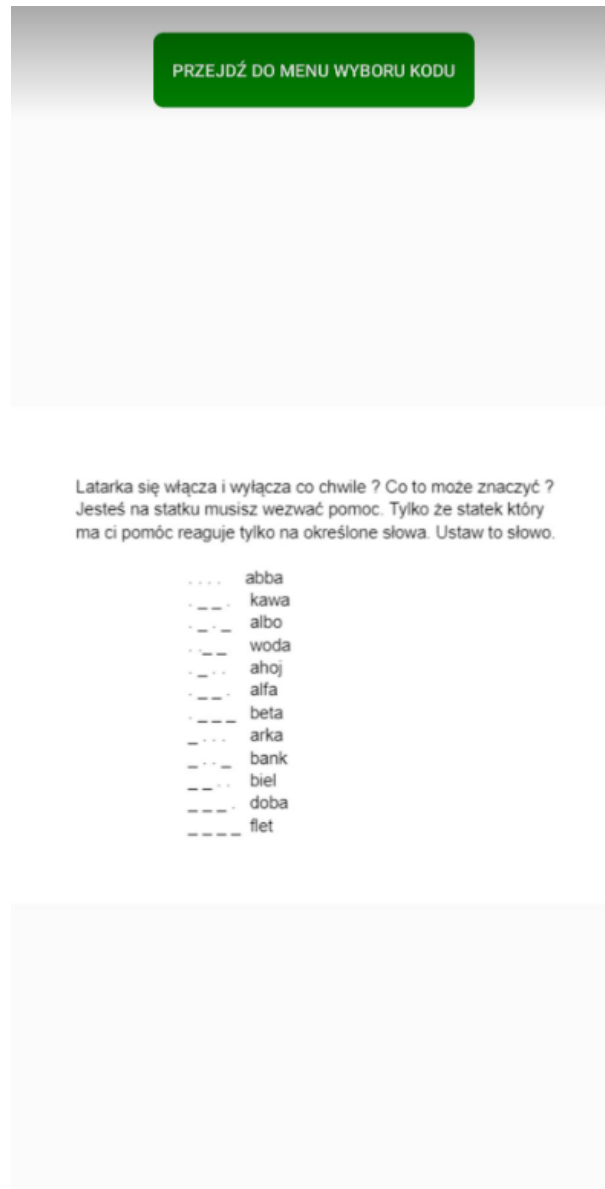
Rys. 6.10. Komunikat (Kod Morsa)

15. Przycisk, którym odtwarzamy kod morsa za pomocą latarki,
16. Przycisk, którym zatwierdzamy hasło,
17. Przyciski pozwalające na zmianę litery na odpowiednim miejscu.



Rys. 6.11. Kod Morsa

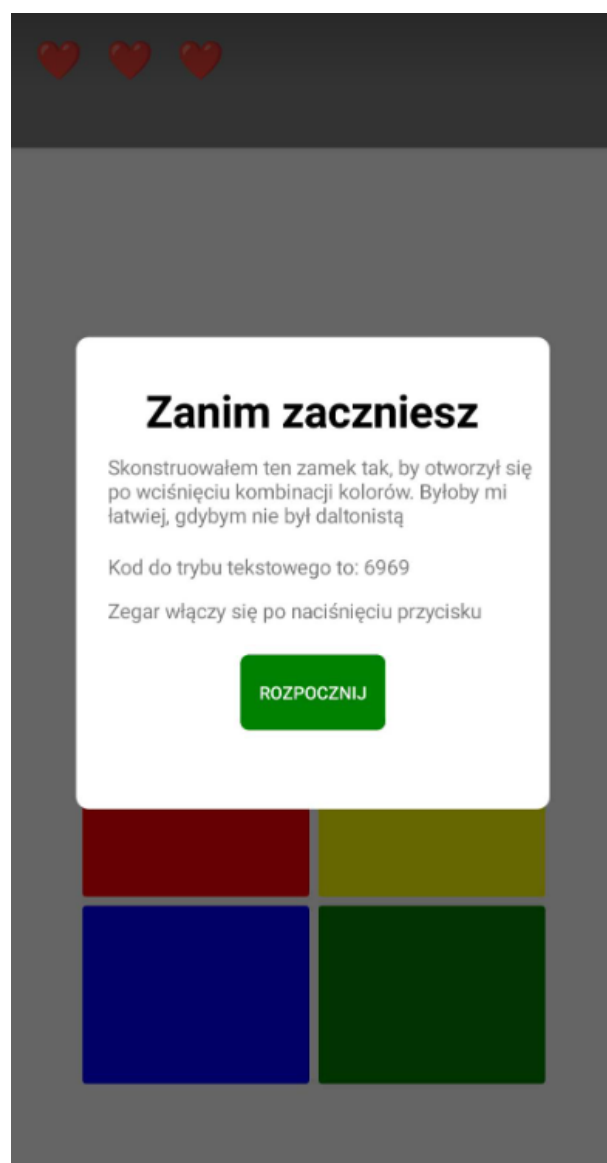
W trybie tekstowym (rys 6.12) dla trybu "Kod Morsa" są rozpisane słowa wraz z ich odpowiednikiem w kodzie. Kropka oznacza sygnał krótki a kreska sygnał długi.



Rys. 6.12. Tryb tekstowy (Kod Morsa)

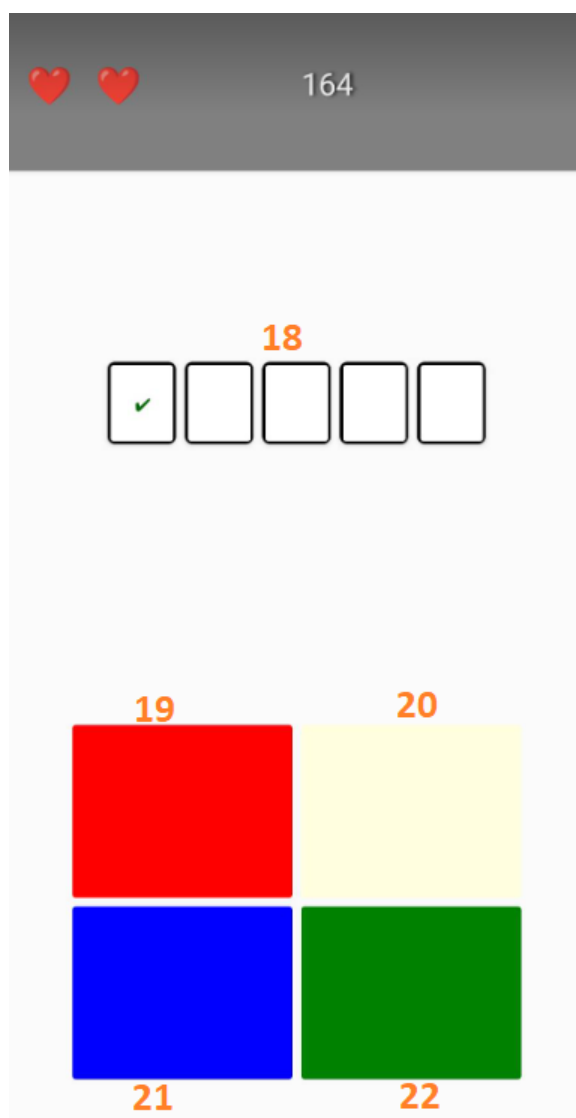
6.6. Kolorki

Po ukończeniu "Kodu Morsa" zostajemy przeniesieni do zagadki, która nazywa się "Kolorki". Podobnie jak w przypadku poprzednich zagadek dostajemy opis zadań i kod do trybu tekstowego (rys 6.13). Ta zagadka polega na odtworzeniu sygnału, który zostaje nam wyświetlony. W trybie tekstowym mamy opisane, przycisk w jakim kolorze mamy nacisnąć po otrzymaniu konkretnego koloru. Gracz w trybie graficznym musi powtórzyć otrzymany kod. Początkowo jeden przycisk stopniowo zwiększając ich liczbę do pięciu.



Rys. 6.13. Komunikat (Kolorki)

18. Licznik, który pokazuje ile jeszcze kolorów nam zostało do zgadnięcia,
19. Przycisk, który odpowiada za określony kolor,
20. Przycisk, który został wyróżniony i gracz musi kliknąć jego odpowiednik na podstawie informacji z trybu tekstowego,
21. Przycisk, który odpowiada za określony kolor,
22. Przycisk, który odpowiada za określony kolor.



Rys. 6.14. Kolorki

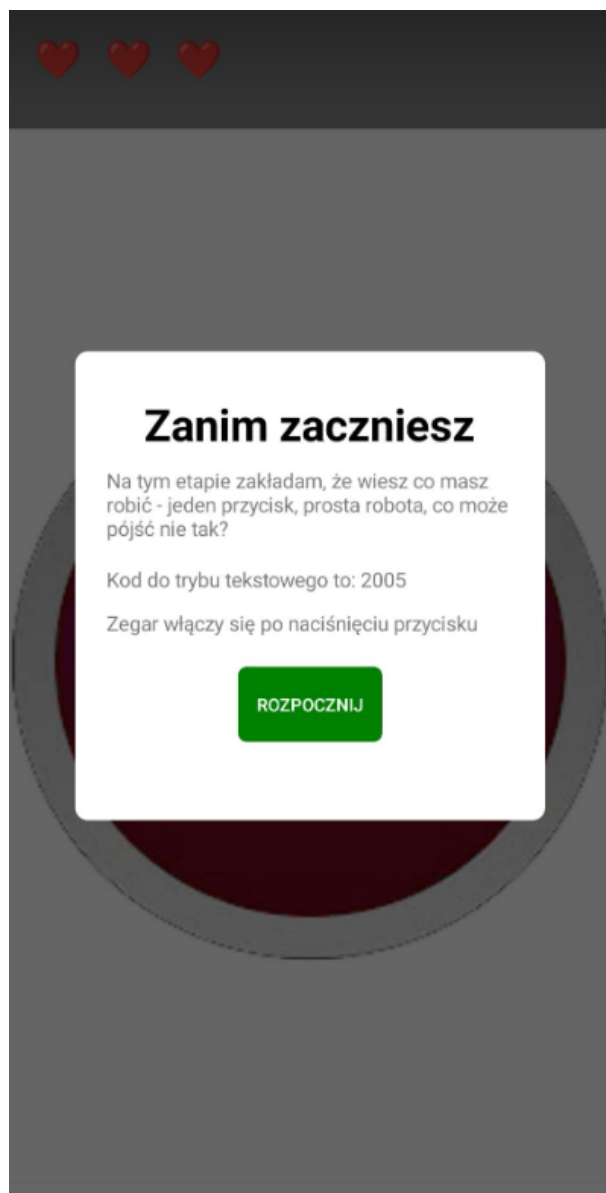
W trybie tekstowym dla tej zagadki (rys. 6.15) jest krótka historyjka i opis, który kolor ma kliknąć gracz w trybie graficznym jeżeli wyświetli się konkretny. Dla przykładu jeżeli gracz z trybu graficznego zobaczy wyróżniony zielony kolor musi wciisnąć czerwony przycisk



Rys. 6.15. Tryb tekstowy (Kolorki)

6.7. Wielki Przycisk

Po ukończeniu "Kolorków" zostajemy przeniesieni do zagadki, która nazywa się "Wielki Przycisk". Podobnie jak w przypadku poprzednich zagadek dostajemy opis zadań i kod do trybu tekstowego (rys 6.16). Ta zagadka polega na przytrzymaniu palca na przycisku przez określony czas zaczynając od określonego czasu na timerze. Wszystko zostało opisane w trybie tekstowym.



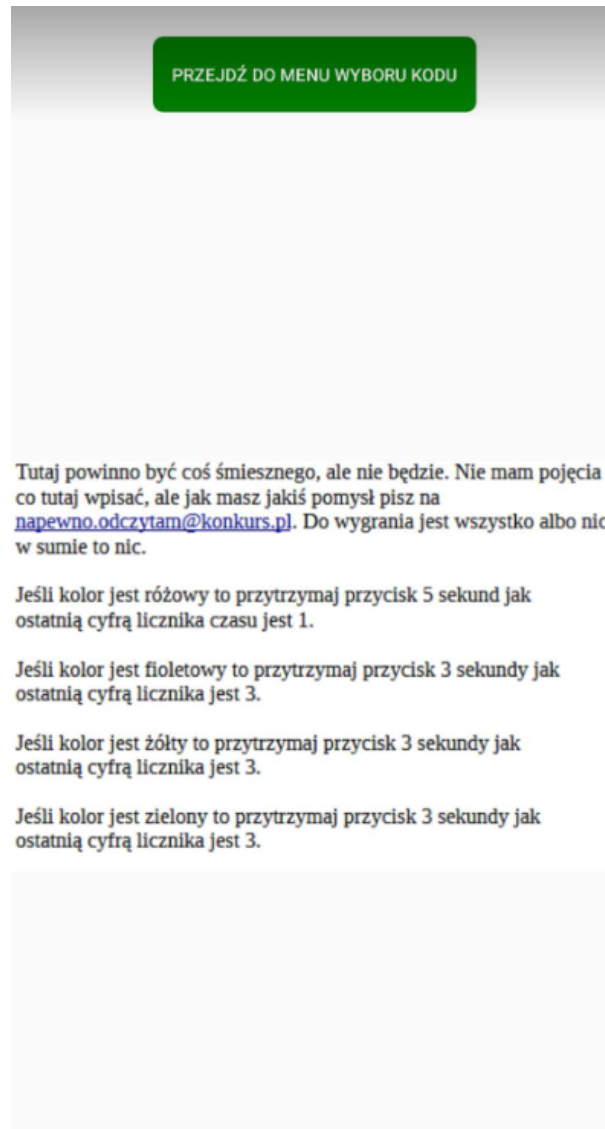
Rys. 6.16. Komunikat (Wielki Przycisk)

23. Wielki Przycisk, który mamy przytrzymać przez określoną liczbę sekund.



Rys. 6.17. Wielki Przycisk

W trybie tekstowym dla zagadki "Wielki Przycisk" (rys 6.18) mamy napisane ile sekund i przy jakim czasie na timerze mamy przytrzymać przycisk.



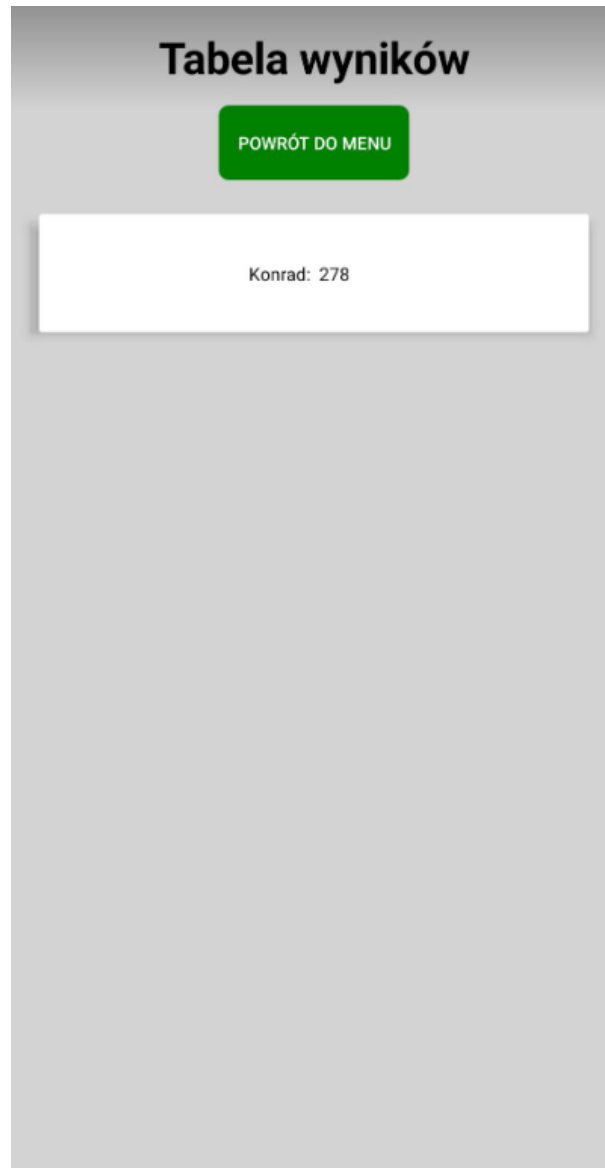
Rys. 6.18. Tekstowy (Wielki Przycisk)

6.8. Tabela wyników

Po przejściu wszystkich zagadek automatycznie przechodzimy do tabeli wyników i w odpowiednim polu wpisujemy swoje imię (rys 6.19). Po podaniu imienia nasz wynik zostaje dodany do tabeli wyników i zestawiony z innymi (rys 6.20).

The image shows a mobile application interface. At the top, the title 'Tabela wyników' is displayed in bold black text. Below it is a green button with the text 'POWRÓT DO MENU'. In the center, there is a white rounded rectangle containing the text 'Podaj imię' in bold black font. Below this text is a text input field with the placeholder 'Podaj imię'. At the bottom of the white rectangle is a green button with the text 'ZATWIERDŹ'.

Rys. 6.19. Tabela wyników (Podaj imię)



Rys. 6.20. Tabela wyników

Spis rysunków

1.1. Labirynt	5
1.2. Literaki	6
1.3. Kod - Tryb graficzny	7
1.4. Kod - Tryb tekstowy	7
1.5. Labirynt	8
1.6. Labirynt	10
3.1. Xamarin	15
3.2. Xamarin Community ToolKit	16
3.3. Visual	17
3.4. GitHub	18
3.5. C#	19
3.6. Komunikat błędu	19
3.7. Menu Główne	21
3.8. Menu Główne v2	22
3.9. Ustawienia	23
3.10. Labirynt	24
3.11. Literaki	25
3.12. Kod morsa	26
3.13. Kolorki	27
3.14. Wielki Przycisk	28
6.1. Menu Główne	46
6.2. Menu Główne v2	47
6.3. Tryb tekstowy	48
6.4. Komunikat (Labirynt)	49
6.5. Labirynt	50
6.6. Tryb tekstowy (Labirynt)	51
6.7. Literaki	52
6.8. Komunikat (Literaki))	53
6.9. Tryb tekstowy (Literaki)	54
6.10. Komunikat (Kod Morsa)	55
6.11. Kod Morsa	56

6.12. Tryb tekstowy (Kod Morsa)	57
6.13. Komunikat (Kolorki)	58
6.14. Kolorki	59
6.15. Tryb tekstowy (Kolorki)	60
6.16. Komunikat (Wielki Przycisk)	61
6.17. Wielki Przycisk	62
6.18. Tekstowy (Wielki Przycisk)	63
6.19. Tabela wyników (Podaj imie)	64
6.20. Tabela wyników	65

Spis tabel

5.1.	42
----------------	----

Spis listingów

1.	Button	28
2.	Label	29
3.	ImageButton	30
4.	MainPage	31
5.	OnAppearing	31
6.	Animate_pulse	32
7.	FirstTask	33
8.	WrongButton	33
9.	HUD	34
10.	Literaki	35
11.	SetTime	36
12.	SetColumns	37
13.	RandomizeWordTable	37
14.	RandomizeText	38
15.	MoveUp	38
16.	ChangeBlockText	39
17.	CheckIfWin	40
18.	MoveDown	40