Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАИП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

**ИГРОВОЕ ПРОГРАММНОЕ СРЕДСТВО “БУНКЕР”**

БГУИР КП 6-05-0612-01 029 ПЗ

Студент гр. 451004 Редько А.М.

Руководитель Данилова Г.В.

Минск 2025

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc198562140)

[1 Анализ предметной области 6](#_Toc198562141)

[1.1 Обзор аналогов 6](#_Toc198562142)

[1.2 Постановка задачи 8](#_Toc198562143)

[2 Проектирование программного средства 9](#_Toc198562144)

[2.1 Структура программы 9](#_Toc198562145)

[2.2 Проектирование интерфейса программного средства 9](#_Toc198562146)

[2.3 Проектирование функционала программного средства 15](#_Toc198562147)

[3 Разработка программного средства 18](#_Toc198562148)

[3.1 Структура карточной игры 18](#_Toc198562149)

[3.2 Визуальные эффекты и анимация 19](#_Toc198562150)

[3.3 Игровая логика и механики 19](#_Toc198562151)

[3.4 Финальная стадия игры 20](#_Toc198562152)

[4 Тестирование программного средства 24](#_Toc198562153)

[5 Руководство пользователя 26](#_Toc198562154)

[5.1 Интерфейс программного средства 26](#_Toc198562155)

[5.2 Управление программным средством 28](#_Toc198562156)

[Заключение 30](#_Toc198562157)

[Список используемых источников 31](#_Toc198562158)

[Приложение A. Исходный код 32](#_Toc198562159)

# ВВЕДЕНИЕ

С момента появления первых компьютерных игр разработчики стремились создавать не только развлекательные, но и интеллектуально насыщенные продукты, которые бы стимулировали мышление, логику и стратегическое планирование. История игровой индустрии богата примерами простых, но увлекательных игр, которые стали классикой благодаря своей доступности и уникальным механикам. Такие игры, как “Tetris”, “Pac-man”, “Змейка” и другие, не только развлекали пользователей, но и способствовали формированию определенного типа мышления, связанного с решением задач и анализом ситуаций. В этом контексте особое место занимают настольные игры, которые, будучи перенесенными в цифровой вид, предлагают новые возможности.

Одной из таких игр является “Бункер” – настольная игра, которая сочетает в себе элементы стратегии, логики и социального взаимодействия. Игра “Бункер” представляет собой дискуссионную карточную игру с постапокалиптическим сюжетом, где игрокам необходимо принимать решения, основываясь на своих ролях, характеристиках и ограниченных ресурсах. Уникальность игры заключается в ее социальной составляющей: игроки должны не только анализировать ситуацию, но и взаимодействовать друг с другом, договариваться, убеждать и даже обманывать, чтобы достичь своей цели. Эта игра стала популярной благодаря своей простоте, глубине и возможности играть как в небольшой компании, так и в больших группах.

Целью данного курсового проекта является разработка цифровой версии игры “Бункер”, которая сохранит все ключевые элементы оригинальной настольной игры, но при этом будет адаптирована для современных платформ. В процессе разработки планируется уделить особое внимание пользовательскому интерфейсу, чтобы сделать игру максимально интуитивно-понятной и удобной для игроков.

Актуальность данной работы обусловлена нестихающим интересом к настольным играм в цифровом формате, особенно в условиях, когда удаленное взаимодействие становится все более востребованным. Игра “Бункер” не только предоставляет возможность для развлечения, но и способствует развитию навыков коммуникации, дискуссии и стратегического мышления.

Таким образом, разработка цифровой версии игры “Бункер” представляет собой не только техническую задачу, но и возможность создать продукт, который будет способствовать развитию социальных и когнитивных навыков у игроков. В процессе работы будут рассмотрены основные аспекты разработки игр, включая проектирование игрового процесса, создание пользовательского интерфейса, реализацию многопользовательского режима и тестирование. Результатом проекта станет готовое игровое программное средство, которое сможет стать достойной альтернативой оригинальной настольной игре и привлечь внимание как любителей настольных игр, так и новых пользователей.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## **Обзор аналогов**

В настоящее время существует множество цифровых адаптаций настольных игр, каждая из которых имеет свои особенности реализации, преимущества и недостатки. Популярность таких игр объясняется удобством онлайн-взаимодействия, автоматизацией игровых процессов и возможностью играть с людьми по всему миру.

Одним из наиболее распространенных способов переноса настольных игр в цифровой формат является использование платформ для создания многопользовательских игр. Эти платформы предоставляют инструменты для эмуляции физических игровых процессов, включая перемещение карт, броски кубиков и взаимодействие между игроками. Например, в “Tabletop Simulator” игроки могут создавать собственные модификации настольных игр, что делает его универсальным решением для любителей жанра. Однако такие платформы требуют от пользователей определенных технических навыков, а также не всегда обеспечивают удобный интерфейс для конкретных игр, что может усложнять процесс игры.

Данное программное средство представлено на рисунке 1.1.

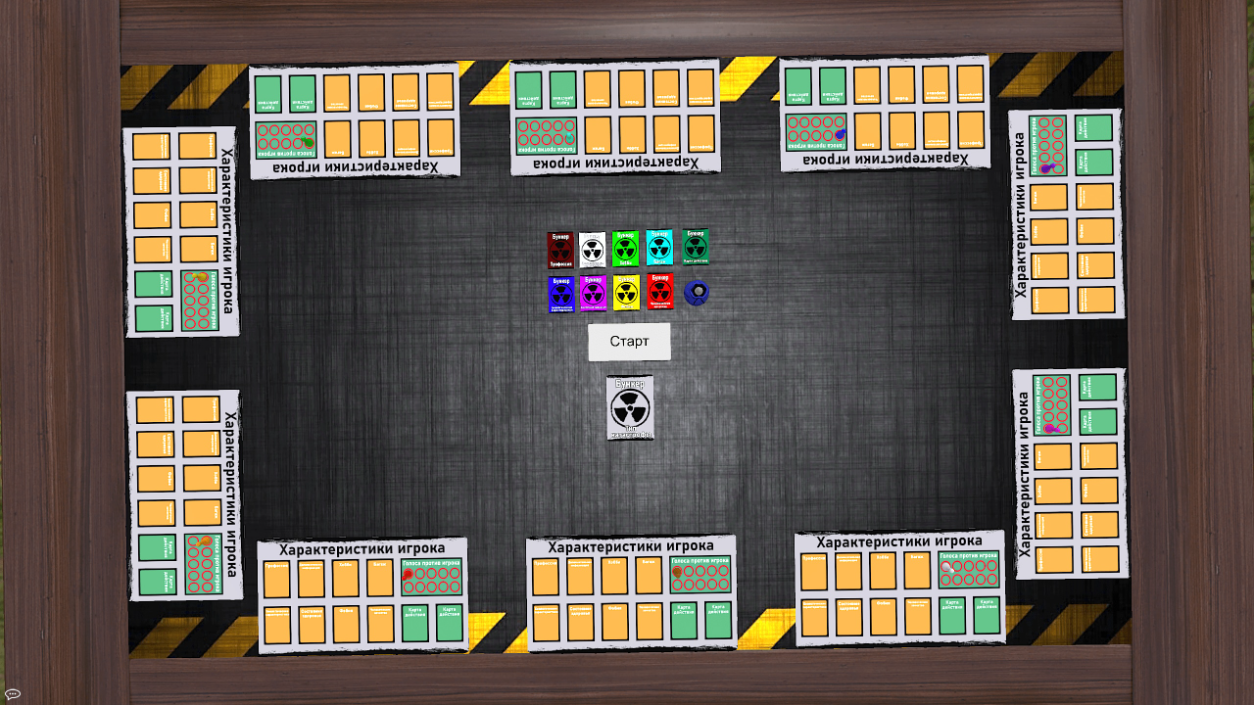


Рисунок 1.1 – Компьютерная игра “Tabletop Simulator”

В настоящее время цифровые адаптации настольных игр активно развиваются, предлагая игрокам новые форматы взаимодействия. Одним из наиболее известных аналогов является игра “Resistance: Avalon”, которая, подобно “Бункеру”, строится на механике социальной дедукции и коллективного принятия решений. Эта реализация демонстрирует удачный пример адаптации настольной игры в цифровой формат, сохраняя при этом ключевые игровые механики.

Данное программное средство представлено на рисунке 1.2.



Рисунок 1.2 – Компьютерная игра “Resistance: Avalon”

Главным преимуществом цифровой версии “Resistance: Avalon” стала автоматизация процессов, которые в настольном варианте требуют участия ведущего. Программа самостоятельно распределяет роли, контролирует соблюдение правил и подсчитывает результаты голосований, что значительно ускоряет игровой процесс. Особого внимания заслуживает реализация сетевого мультиплеера, позволяющая игрокам со всего мира участвовать в совместных партиях через интернет. Встроенные инструменты коммуникации, включая текстовый и голосовой чат, помогают сохранить элемент социального взаимодействия, который является важной составляющей подобных игр.

Главным недостатком существующих аналогов является отсутствие специализированных решений для игры “Бункер”. Большинство платформ либо слишком универсальны, либо не учитывают специфику этой игры, включая ее механики взаимодействия, систему ролей и процесс голосования.

## **Постановка задачи**

В рамках данной курсовой работы будет разработана цифровая версия настольной игры “Бункер”. Основной целью проекта будет сохранение всех ключевых механик оригинальной игры, обеспечивая при этом удобный интерфейс и автоматизацию игровых процессов.

В программном средстве будут реализованы следующие функции:

– создание новой игровой сессии с настройкой количества игроков;

– генерация уникальных характеристик для каждого игрока;

– загрузка и сохранение текущего состояния игры;

– просмотр карт всех участников с их характеристиками;

– пошаговое раскрытие информации о персонажах;

– система исключения игроков;

– обработка решений по ситуациям в бункере;

– автоматический расчет шансов на выживание;

– анимированное отображение результатов игры;

– просмотр правил и подсказок в процессе игры.

Для разработки программного средства будет использоваться язык программирования Delphi и среда разработки Embarcadero Delphi 12.0 Community Edition.

# ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## **Структура программы**

При разработке приложения будет использовано семь модулей:

* StartUnit – отвечает за отображение начального меню приложения.
* RuleUnit – содержит полное описание игрового процесса, механик, правил и особенностей игры.
* SeedEnterUnit – отвечает за систему генерации игровых сессий через уникальные ключи.
* PreparationUnit – является настройками игровой сессии.
* MainUnit – отвечает за отображение игрового поля, карт персонажей и управление игровым процессом.
* FinalUnit – отвечает за расчет и отображение результатов, анимацию победы/поражения.
* ConnectUnit – выступает промежуточным звеном между этапом подготовки и основным игровым процессом.

## **Проектирование интерфейса программного средства**

Дизайн программных продуктов является ключевым фактором, влияющим на удобство взаимодействия пользователей и общее качество программного средства. Для достижения этих целей интерфейс должен соответствовать следующим базовым принципам:

* оптимизация рабочего пространства;
* интуитивная понятность.

**2.2.1** Начальное окно

Начальное окно приложения состоит из четырех компонентов TLabel и двух кнопок компонента TMainMenu, с помощью которых пользователь может создать новую игру, подключиться по ключу игры, ознакомиться с правилами, ознакомиться с разработчиком, загрузить сохраненную игру из файла, либо осуществить выход:

* нажатие на надпись “Новая игра” открывает модуль PreparationUnit, где пользователь начинает настройку игровой сессии;
* нажатие на надпись “Подключиться” открывает модуль SeedEnterUnit, где пользователь может ввести уникальный ключ;
* нажатие на надпись “Правила” отобразит модуль RuleUnit, где пользователь может ознакомиться с правилами;
* нажатие на надпись “Выход” закрывает приложение.

Макет начального окна представлен на рисунке 2.1.

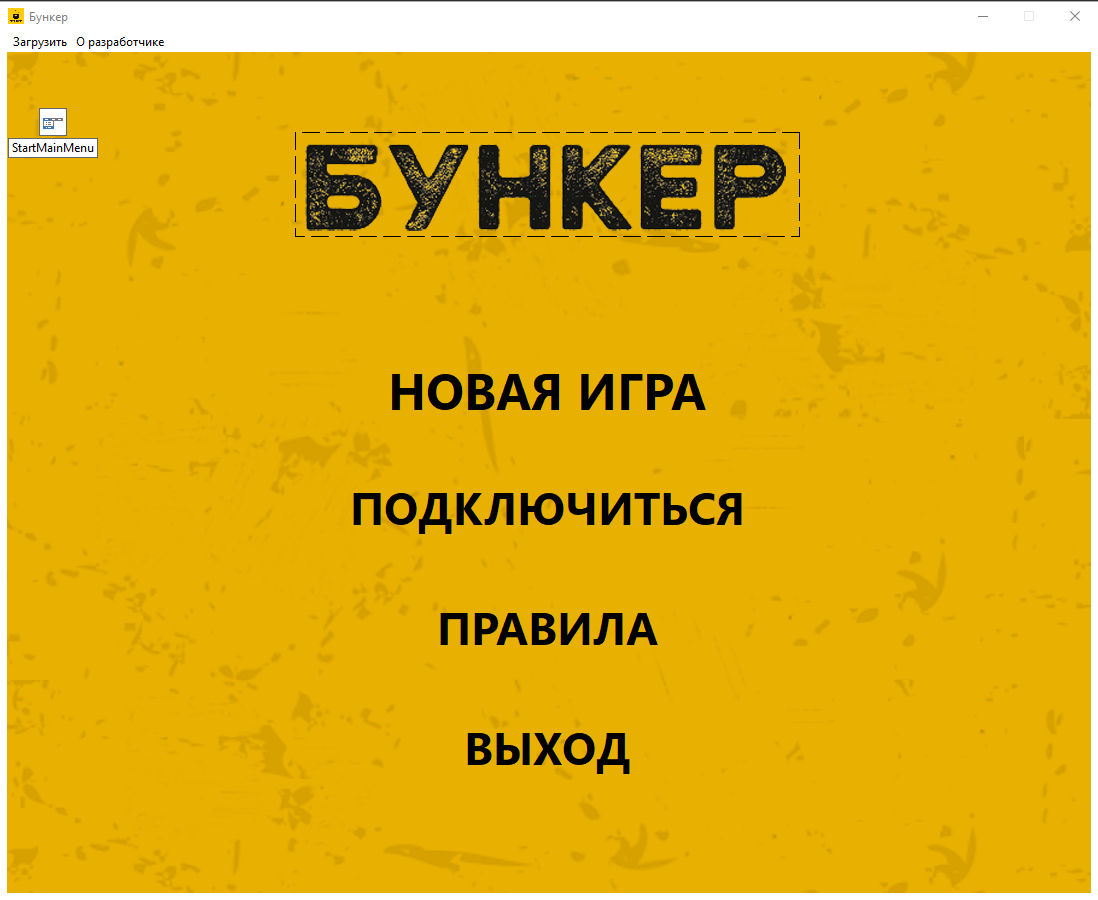


Рисунок 2.1 – Макет начального окна

**2.2.2** Окно с правилами

Окно с правилами состоит из трех компонентов TImage. При нажатии на любой из них компонент скрывается и отображает следующую страницу правил.

Макет окна с правилами представлен на рисунке 2.2.

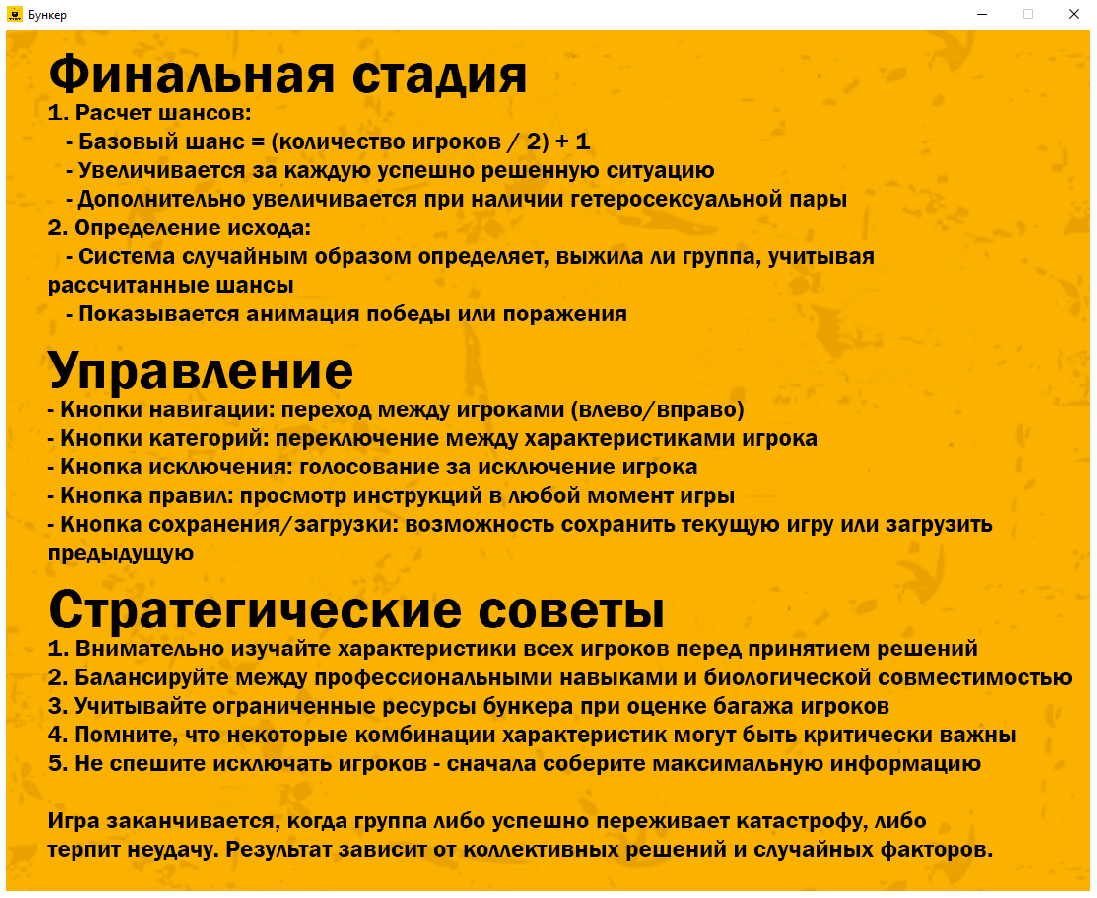


Рисунок 2.2 – Макет окна с правилами

**2.2.3** Окно ввода ключа

Окно ввода ключа состоит из пятнадцати компонентов TImage и двух компонентов TLabel. При нажатии на цифры вводится уникальный ключ, а при нажатии на кнопку "Стереть" удаляется последняя введенная цифра. Кнопка "Назад" возвращает в главное меню, а кнопка "Далее" открывает окно выбора персонажа.

Макет окна ввода ключа представлен на рисунке 2.3.



Рисунок 2.3 – Макет окна ввода ключа

**2.2.4** Окно подготовки

Окно подготовки состоит из трех компонентов TImage и трех компонентов TLabel. При нажатии кнопок "Влево" и "Вправо" количество игроков уменьшается или увеличивается соответственно.

Макет окна подготовки представлен на рисунке 2.4.



Рисунок 2.4 – Макет окна подготовки

**2.2.5** Главное окно

Главное окно состоит из десяти компонентов TImage, шести компонентов TLabel, двух компонентов TTimer, трех кнопок компонента TMainMenu, двух компонентов TVirtualImageList и двух компонентов TImageCollection.

Главное окно приложения представляет собой основной игровой интерфейс, отображаемый после начала игровой сессии. В центре окна расположена карточка игрока с текстовым описанием. Слева и справа от карточки находятся кнопки переключения между категориями информации.

В верхней части интерфейса расположен элемент TLabel, отображающий номер текущего игрока. Для навигации между игроками используются кнопки–стрелки. Снизу находится кнопка исключения игрока, а в левом нижнем углу – кнопка возврата в главное меню.

Кнопка справки активирует отображение таблицы правил. Кнопка перехода к финалу появляется автоматически при выполнении условий завершения игры.

Реализована система сохранения и загрузки игрового прогресса через стандартное меню "Файл".

Макет главного окна представлен на рисунке 2.5.

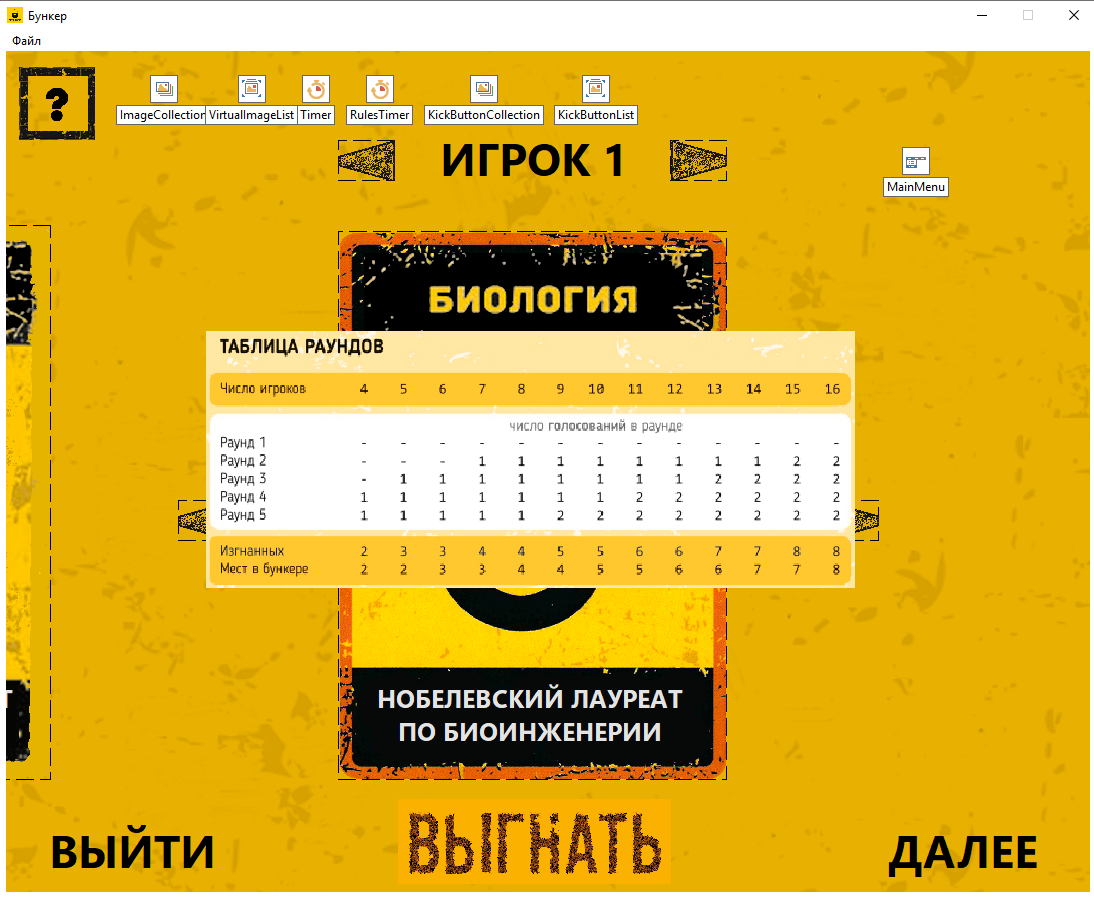


Рисунок 2.5 – Макет главного окна

**2.2.6** Финальное окно

Финальное окно состоит из восьми компонентов TImage, четырех компонентов TLabel и компонента TTimer.

Функциональность финального окна реализована через анимационную последовательность, которая включает несколько этапов:

* появление вопроса с анимированными кнопками ответа;
* отображение информационного сообщения с эффектом постепенного набора текста;
* последовательный показ игровых ситуаций с возможностью выбора ответов;
* расчет и отображение шансов на победу;
* финальную анимацию с затемнением и показом результата.

Макет главного окна представлен на рисунке 2.6.

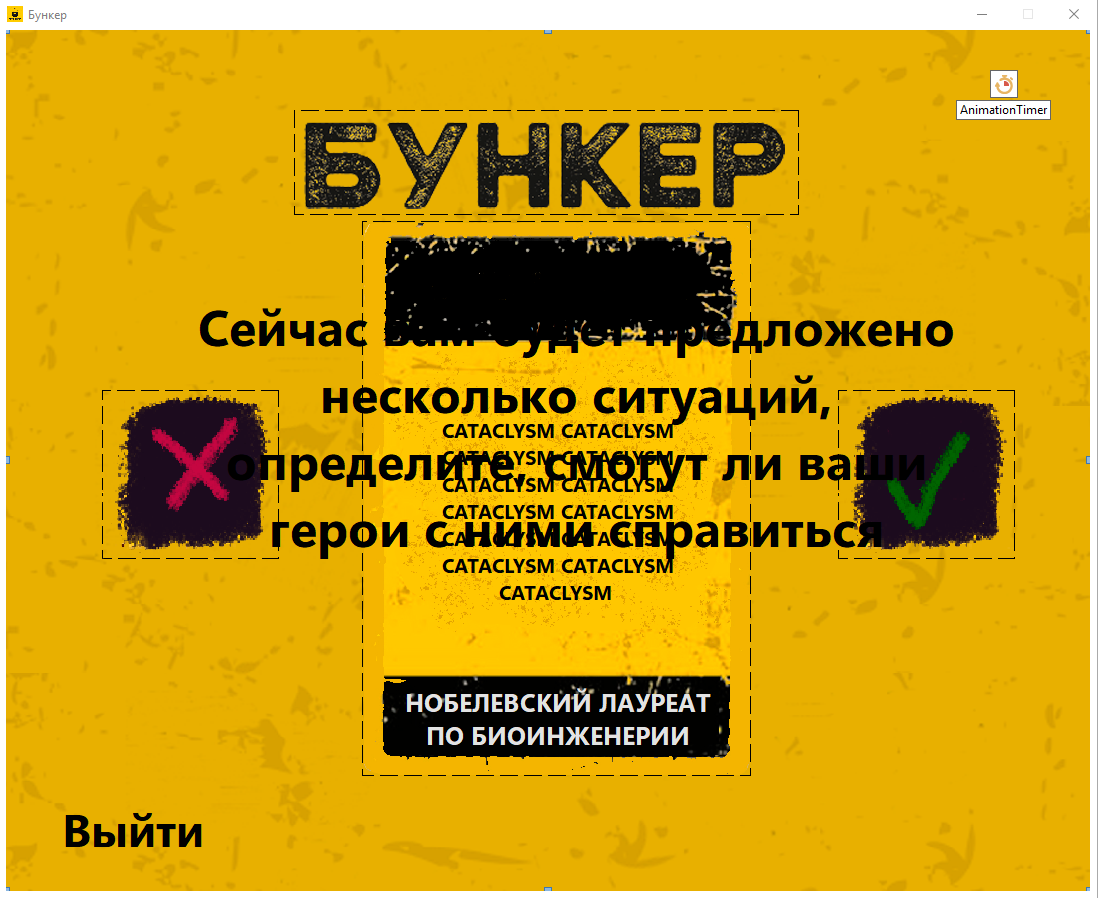


Рисунок 2.6 – Макет финального окна

**2.2.7** Окно подключения

Окно подключения состоит из пяти компонентов TImage и шести компонентов TLabel. В окне отображается ключ игры для подключения, а также происходит выбор активного игрока.

Макет главного окна представлен на рисунке 2.7.



Рисунок 2.7 – Макет окна подключения

## **Проектирование функционала программного средства**

Программное средство представляет собой интерактивную игру, которая моделирует процесс выживания группы людей после глобальной катастрофы. Основные функциональные модули включают управление игровыми картами, обработку игрового процесса и работу с сохранениями.

**2.3.1** Управление игровыми картами

В модуле CardSystem реализована генерация и управление игровыми картами различных категорий: профессии, биологические характеристики, состояние здоровья, хобби, багаж, особые факты, катастрофы и ситуации.

Генерация карт происходит при инициализации игры через процедуру GenerateGameSetup.

Ниже приведена блок–схема процедуры GenerateGameSetup на рисунке 2.8.



Рисунок 2.8 – Блок-схема процедуры GenerateGameSetup

**2.3.2** Загрузка ситуаций бункера

Загрузка ситуаций бункера происходит из основного игрового набора FGameSetup в связный список для последующего использования в финальной части игры.

Ниже приведена блок–схема процедуры LoadSituationsFromGameSetup на рисунке 2.9.



Рисунок 2.9 – Блок-схема процедуры LoadSituationsFromGameSetup

**2.3.3** Сохранение игры

Система сохранения игры реализована через структуру данных TGameSave, содержащую текущее состояние игры.

Блок–схема процедуры сохранения сессии в файл отображена на рисунке 2.10.



Рисунок 2.10 – Блок-схема процедуры SaveDataFile

Эти процедуры реализуют все основные задачи игрового программного средства.

# РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## **Структура карточной игры**

Целью программы является создание интерактивной карточной игры "Бункер", где игроки принимают решения в постапокалиптическом сценарии. Основной акцент сделан на взаимодействии между игроками и обработке их решений.

Процедура инициализации карт игроков представлена ниже:

// Инициализируются карты игроков

Procedure InitializePlayersCards;

Var

I: Integer;

Cat: TCardCategory;

Begin

// Выделяется память под массив карт игроков

SetLength(FPlayersCards, GameSetup.Players);

// Перебираются все игроки

For I := 0 To GameSetup.Players - 1 Do

Begin

// Устанавливается, что карта не раскрывается

FPlayersCards[I].Revealed := False;

// Устанавливается, что игрок не выбывает

FPlayersCards[I].Kicked := False;

// Очищаются тексты карт для всех категорий

For Cat := TCardCategory.First To TCardCategory.Last Do

Begin

FPlayersCards[I].CardText[Cat] := '';

End;

End;

// Проверяется, есть ли игроки

If GameSetup.Players > 0 Then

Begin

// Устанавливается, что карта текущего игрока раскрывается

FPlayersCards[CurrentPlayerIndex].Revealed := True;

// Заполняются тексты карт для всех категорий текущего игрока

For Cat := TCardCategory.First To TCardCategory.Last Do

Begin

FPlayersCards[CurrentPlayerIndex].CardText[Cat] :=

GameSetup.Cards[GetCardTypeFromCategory(Cat)

][CurrentPlayerIndex].Description;

End;

End;

End;

Данная процедура автоматизирует процесс раздачи карт. После базовой инициализации система заполняет карты текущего игрока реальными значениями из игрового набора. Это позволяет сразу отобразить информацию для первого игрока, в то время как карты остальных остаются скрытыми.

## **Визуальные эффекты и анимация**

Для улучшения пользовательского опыта в игре реализована система плавных переходов между состояниями. Основные визуальные эффекты включают: плавное появление/исчезание элементов, анимацию изменения размеров карт, затемнение исключенных игроков. Система анимации построена на основе таймера и управляется через параметры прозрачности.

Ниже представлен код эффекта перелистывания карт:

Procedure StartAnimation(TargetIndex: Integer);

Begin

FTargetIndex := TargetIndex; // Целевое состояние

PrepareBitmaps; // Подготовка изображений

FAlpha := 0; // Начальное значение прозрачности

Timer.Enabled := True; // Запуск анимации

// Эффект уменьшения карты перед переходом

Card.Width := Card.Width Div 2;

Card.Left := Card.Left + Card.Width Div 2;

End;

Каждый кадр анимации пересчитывает параметры отображения и применяет их к графическим элементам. Это создает эффект плавного перехода между различными состояниями интерфейса.

## **Игровая логика и механики**

Основной игровой процесс строится вокруг нескольких ключевых механик.

**3.3.1** Просмотр характеристик игроков

Игроки могут поочередно открывать карты друг друга, чтобы получить информацию для принятия решений. Реализация включает проверку состояния карты перед отображением.

Ниже представлена процедура открытия карт:

Procedure ToggleCardReveal;

Var

PlayerIndex: Integer;

Begin

PlayerIndex := GetPlayerIndex;

// Открывается карта только если она была скрыта

If Not FPlayersCards[PlayerIndex].Revealed Then

Begin

FPlayersCards[PlayerIndex].Revealed := True;

// Загружается описание из игрового набора

FPlayersCards[PlayerIndex].CardText[FCurrentCategory] :=

GameSetup.Cards[GetCardTypeFromCategory(FCurrentCategory)

][PlayerIndex].Description;

ShowCurrentCard; // Обновляется отображение

End;

End;

**3.3.2** Исключение игроков

Важнейший элемент стратегии – возможность исключать игроков из бункера. Этот процесс включает: подтверждение действия через диалоговое окно, визуальное обозначение исключенного игрока и обновление интерфейса.

Ниже представлена процедура исключения игрока:

Procedure KickButtonClick(Sender: TObject);

Var

PlayerIndex: Integer;

Begin

PlayerIndex := GetPlayerIndex;

// Запрос подтверждения действия

If MessageBox(Handle, 'Вы действительно хотите выгнать игрока?',

'Подтверждение', MB\_YESNO Or MB\_ICONQUESTION) = IDYES Then

Begin

FPlayersCards[PlayerIndex].Kicked := True; // Помечается как исключенный

UpdateKickButton; // Обновляется кнопка

KickCardReveal; // Показываются все карты исключенного

End;

End;

Эти процедуры реализуют логику и механику игрового программного средства.

## **Финальная стадия игры**

Когда остается половина игроков, система переходит к финальной стадии, где оцениваются шансы на выживание. Анализируются характеристики оставшихся игроков, проверяется наличие гетеросексуальной пары и генерируется случайное событие с учетом рассчитанных вероятностей.

Ниже представлена процедура подсчета шансов:

Procedure CountTheChance;

Begin

// Учет гетеросексуальной пары

If HasPair Then

Inc(Chance);

// Расчет процентного соотношения

Percent := Round((Chance / MaxChance) \* 100);

// Запуск анимации отображения результата

AnimationStep := 200;

AnimationTimer.Enabled := True;

End;

Визуальное представление результата включает анимированное появление текста и финального изображения.

Ниже представлена процедура, отвечающая за финальную анимацию:

Procedure ShowNextImage;

Var

FadeAlpha: Integer;

FadeBitmap: TBitmap;

MaxChance: Integer;

TheEnd: Integer;

Begin

// Обрабатывается текущий шаг анимации

Case AnimationStep Of

0:

Begin

FadeInImage(QuestionImage);

AnimationTimer.Interval := 600;

Inc(AnimationStep);

End;

1:

Begin

// Проверяется наличие пары

If HasPair Then

Begin

// Устанавливается позиция кнопки "Да"

YesButton.Left := 464;

FadeInImage(YesButton);

End

Else

Begin

// Устанавливается позиция кнопки "Нет"

NoButton.Left := 464;

FadeInImage(NoButton);

End;

AnimationTimer.Interval := 200;

Inc(AnimationStep);

End;

2 .. 19:

Begin

// Мигает соответствующая кнопка в зависимости от наличия пары

If HasPair Then

BlinkImage(YesButton)

Else

BlinkImage(NoButton);

Inc(AnimationStep);

End;

20:

Begin

// Скрываются элементы интерфейса

QuestionImage.Visible := False;

YesButton.Visible := False;

NoButton.Visible := False;

InfoLabel.Caption := '';

InfoLabel.Visible := True;

// Устанавливается полное сообщение

FullMessage :=

'Сейчас вам будет предложено несколько ситуаций,

определите, смогут ли ваши герои с ними справиться';

CurrentTextPos := 0;

// Устанавливается интервал таймера

AnimationTimer.Interval := 50;

// Увеличивается шаг анимации

Inc(AnimationStep);

End;

21 .. 120:

Begin

// Отображается текст посимвольно

If CurrentTextPos < Length(FullMessage) Then

Begin

Inc(CurrentTextPos);

InfoLabel.Caption := Copy(FullMessage, 1, CurrentTextPos);

End

Else If AnimationStep = 120 Then

Begin

// Устанавливается задержка после полного отображения текста

AnimationTimer.Interval := 4000;

End;

Inc(AnimationStep);

End;

121:

Begin

// Обновляются элементы интерфейса для отображения ситуации

InfoLabel.Visible := False;

BunkerSituationImage.Visible := True;

BunkerSituation.Visible := True;

SituationTitle.Visible := True;

// Загружаются ситуации из настроек игры

LoadSituationsFromGameSetup;

// Инициализируется первая ситуация

InitializeSituation(FSituations.First);

// Отображаются кнопки ответа

YesButton.Visible := True;

NoButton.Visible := True;

YesButton.Top := 360;

YesButton.Left := 832;

NoButton.Top := 360;

NoButton.Left := 96;

YesButton.OnClick := YesButtonClick;

NoButton.OnClick := NoButtonClick;

Inc(AnimationStep);

End;

200:

Begin

// Подготавливается метка для отображения шансов

InfoLabel.Caption := '';

InfoLabel.Visible := True;

MaxChance := (FGameSetup.Players Div 2) + 1;

If MaxChance = 0 Then

MaxChance := 1;

// Формируется сообщение о шансах

FullMessage := 'Ваши шансы победить: ' +

IntToStr(Percent) + '%';

CurrentTextPos := 0;

AnimationTimer.Interval := 20;

Inc(AnimationStep);

End;

201 .. 250:

Begin

// Отображается текст посимвольно

If CurrentTextPos < Length(FullMessage) Then

Begin

Inc(CurrentTextPos);

InfoLabel.Caption := Copy(FullMessage, 1, CurrentTextPos);

End

Else If AnimationStep = 250 Then

Begin

AnimationTimer.Interval := 2000;

End;

Inc(AnimationStep);

End;

251:

Begin

// Скрываются элементы интерфейса

InfoLabel.Visible := False;

BunkerSituationImage.Visible := False;

SituationTitle.Visible := False;

BunkerSituation.Visible := False;

YesButton.Visible := False;

NoButton.Visible := False;

QuestionImage.Visible := False;

Zagolovok.Visible := False;

BackButtonLabel.Visible := False;

AnimationTimer.Interval := 150;

Inc(AnimationStep);

End;

252 .. 280:

Begin

// Рассчитывается прозрачность для эффекта затухания

FadeAlpha := (AnimationStep - 251) \* 10;

If FadeAlpha > 255 Then

FadeAlpha := 255;

// В конце анимации отображается результат

If AnimationStep = 280 Then

Begin

// Определяется исход игры случайным образом с учетом шансов

If Random(100) < Percent Then

TheEnd := 1

Else

TheEnd := 0;

// Отображается экран победы или поражения

If TheEnd = 1 Then

Win.Visible := True

Else

Lose.Visible := True;

// Скрываются оставшиеся элементы интерфейса

BunkerSituationImage.Visible := False;

SituationTitle.Visible := False;

BunkerSituation.Visible := False;

YesButton.Visible := False;

NoButton.Visible := False;

InfoLabel.Visible := False;

// Останавливается таймер анимации

AnimationTimer.Enabled := False;

End;

Inc(AnimationStep);

End;

End;

End;

Вышеупомянутые процедуры реализуют финальную часть программного средства.

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Основной проблемой являлись ошибки отображения карт игрока. При переключении категорий отображался некорректный текст или несоответствующая изображению картинка. Проблема была вызвана тем, что процедуры обработки изменяемой информации получали неверные индексы.

Проблема представлена на рисунке 4.1.



Рисунок 4.1 – Проблема с фильтрацией информации

Проблема была решена изменением процедуры отображения карт, которая имеет вид:

Procedure ShowCurrentCard;

Var

PlayerIndex: Integer;

CardTextEmpty: Boolean;

BlendFunc: TBlendFunction;

TempBmp: TBitmap;

Begin

// Получение индекса текущего игрока и проверка, пуст ли текст карты

PlayerIndex := GetPlayerIndex;

CardTextEmpty := (FPlayersCards[PlayerIndex].CardText[FCurrentCategory] = '');

// Обрабатка различных случаев отображения в зависимости от состояния

карты и категории

If (FCurrentCategory = CcProfession) And Not FPlayersCards[PlayerIndex].Revealed

And Not Timer.Enabled Then

Begin

// Случай 1: Категория "Профессия", карта не раскрыта, анимация не активна

FTargetIndex := GetCategoryIndex(FCurrentCategory);

PrepareBitmaps;

StartAnimation(CARD\_BACK\_INDEX);

ProfLabel.Caption := '';

End

Else If (Not FPlayersCards[PlayerIndex].Revealed Or CardTextEmpty)

And Not Timer.Enabled Then

Begin

// Случай 2: Карта не раскрыта или текст пуст, анимация не активна

FTargetIndex := GetCategoryIndex(FCurrentCategory);

PrepareBitmaps;

StartAnimation(CARD\_BACK\_INDEX);

ProfLabel.Caption := '';

End

Else

Begin

// Стандартный случай: Показ карты с соответствующей анимацией и текстом

StartAnimation(GetCategoryIndex(FCurrentCategory));

ProfLabel.Caption := FPlayersCards[PlayerIndex].CardText[FCurrentCategory];

End;

End;

Большинство проблем возникло при работе с TVirtualImageList, на стадии тестирования все проблемы были исправлены.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## **Интерфейс программного средства**

**5.1.1** Начальное окно

Начальное окно программы, изображенное на рисунке 5.1, содержит следующие кнопки:

* “Новая игра”;
* “Подключиться”;
* “Правила”;
* “Выход”.

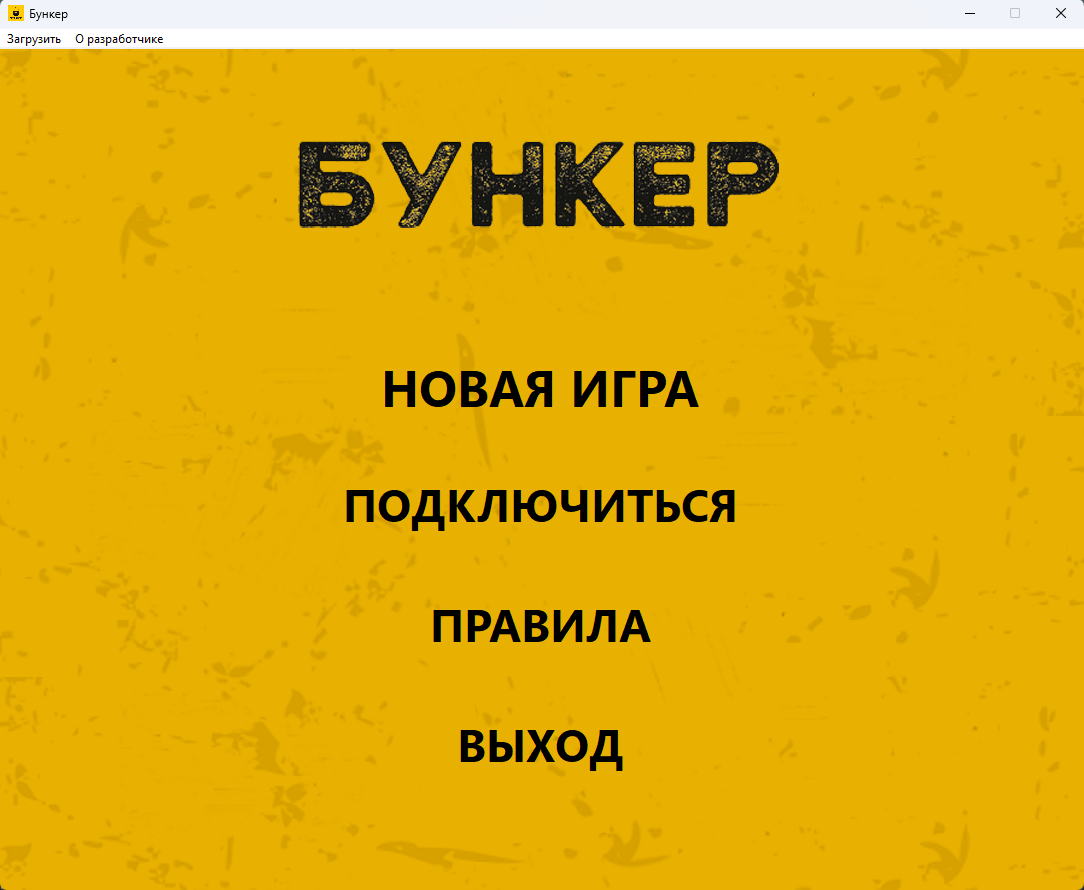


Рисунок 5.1 – Внешний вид начального окна

* + 1. Окна подготовки к игровой сессии

При нажатии на кнопку “Новая игра” пользователю предоставляется окно выбора количества игроков. При нажатии на кнопку “Далее” пользователю предоставляется окно выбора активного игрока. Окна подготовки представлены на рисунке 5.2.

Рисунок 5.2 – Окна подготовки

**5.1.3** Основное окно

После подготовки пользователю представляется окно основной игры, где изображены карты игроков и карта катастрофы. В данном окне осуществляется основной игровой процесс, где пользователи раскрывают карты и исключают игроков. Вид основного окна представлен на рисунке 5.3.

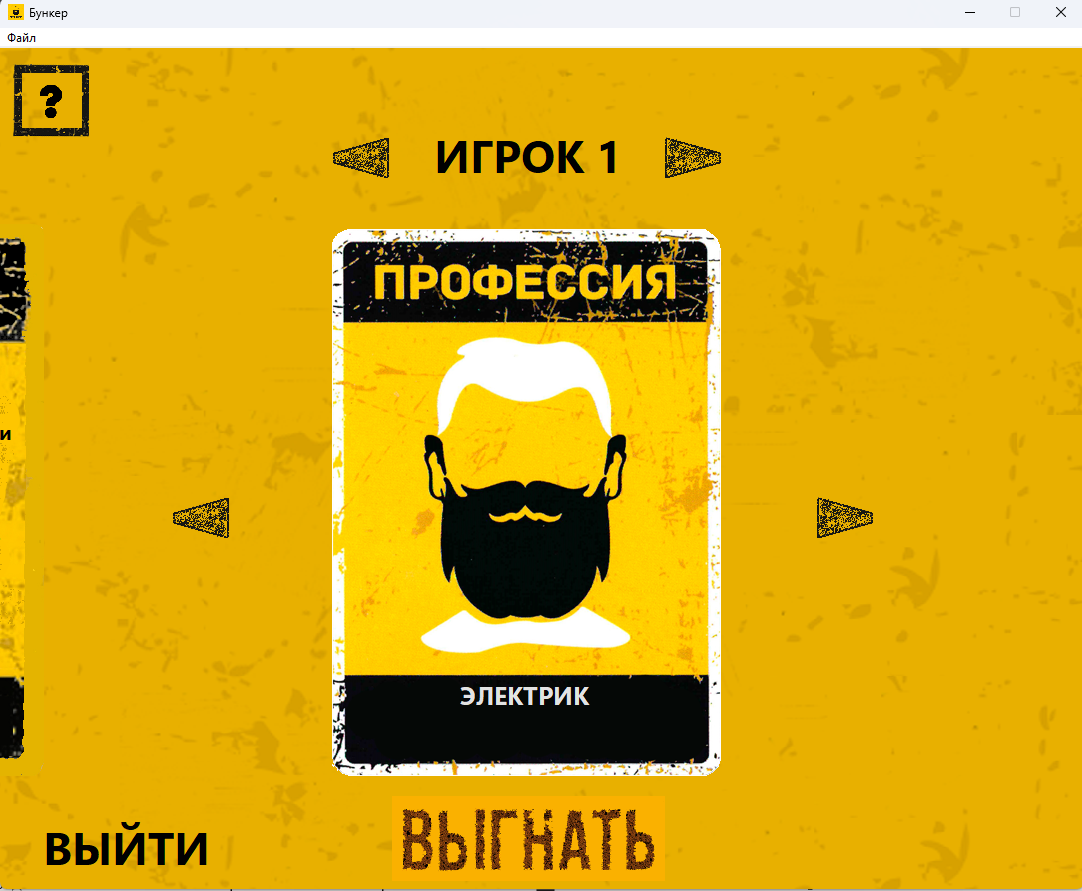


Рисунок 5.3 – Основное окно

**5.1.4** Финальное окно

Когда исключена половина игроков, активируется кнопка "Далее". Ее нажатие запускает финальную стадию игры, включающую анимацию ключевых моментов игры и демонстрацию карт ситуаций. После ответа на все предложенные ситуации система отображает финальный результат – выжили игроки или нет. Интерфейс финального окна представлен на рисунке 5.4.

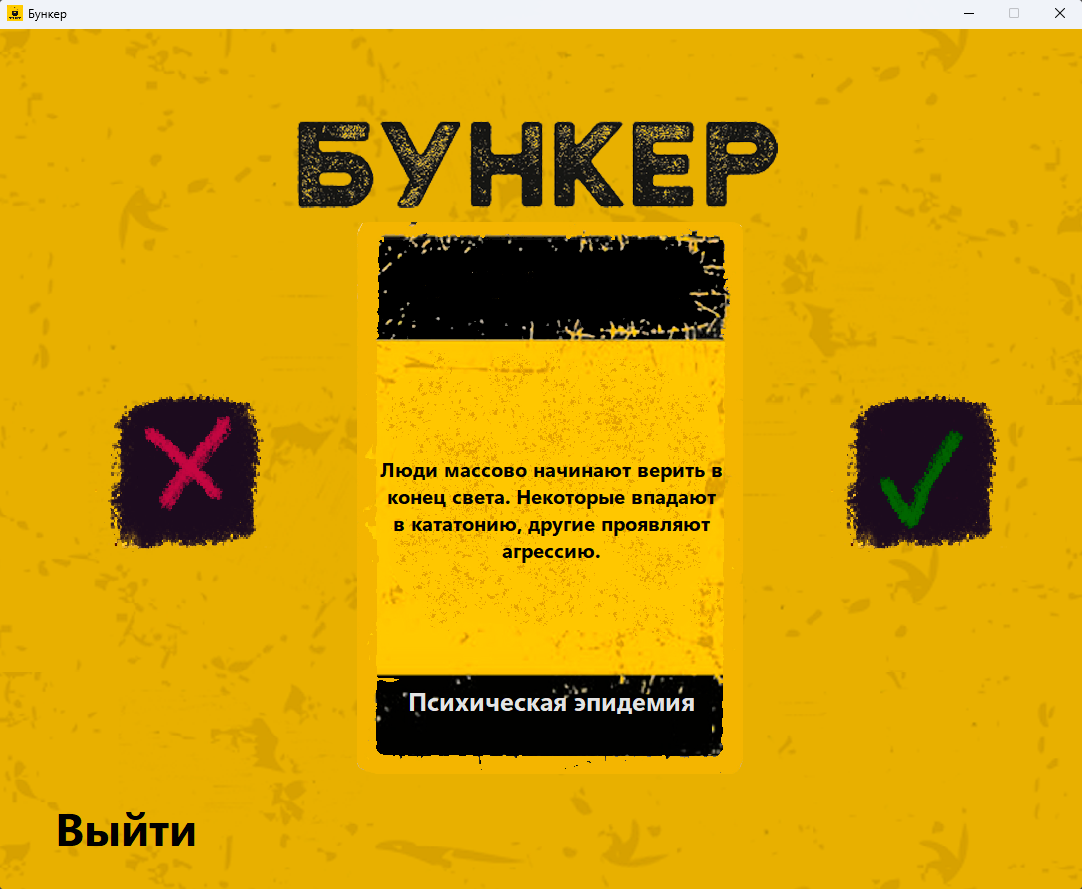


Рисунок 5.4 – Вид финального окна

После анимаций и результатов, пользователь вернется на начальное окно.

## **Управление программным средством**

После запуска программы пользователь может выбрать один из трех вариантов начала работы: загрузить ранее сохраненную игру из файла, начать новую игру с произвольными параметрами или возобновить игру с заранее известными параметрами по ключу.

Для того чтобы загрузить данные из файла, нужно нажать кнопку “Файл” – “Открыть” в меню сверху экрана.

Кнопки “Влево” и “Вправо” сверху экрана позволяют циклически переключаться между игроками. Текущий активный игрок отображается в центральной верхней части экрана “ИГРОК N”.

.

Просмотр категорий карт осуществляется кнопками “Влево” и “Вправо” по бокам от карт игрока. Для открытия карты нужно кликнуть по центральному изображению карты. Название текущей категории отображается под картой.

Для исключения текущего игрока нужно нажать кнопку “Выгнать” внизу экрана. Перед исключением система запросит подтверждение действия.

Для того чтобы переключить страницу правил, нужно нажать на любое место экрана в окне с правилами.

Для того чтобы получить подсказку во время игры, нужно нажать на значок “?” в левом верхнем углу экрана.

Интерфейс разработан с учетом эргономических требований и обеспечивает комфортный игровой процесс как для новичков, так и для опытных пользователей.

# ЗАКЛЮЧЕНИЕ

Результатом выполнения данной курсовой работы стало разработанное программное средство – интерактивная карточная игра “Бункер”, созданная в интегрированной среде разработки Delphi 12.0. В процессе реализации проекта удалось достичь поставленных целей: создать увлекательную игру с удобным интерфейсом, реалистичной механикой принятия решений и комплексной системой оценки шансов на выживание.

В ходе выполнения проекта была проведена значительная работа:

* анализ существующих аналогов и механик настольных игр;
* проектирование гибкой архитектуры с возможностью расширения;
* реализация сложной логики взаимодействия игровых элементов;
* тестирование баланса и пользовательского опыта.

Особое внимание было уделено: пользовательскому интерфейсу, реалистичности механик, визуальному оформлению.

Перспективы развития проекта включают:

* добавление новых типов катастроф и игровых ситуаций;
* реализацию сетевого режима для многопользовательской игры;
* углубление системы характеристик игроков;
* разработку мобильной версии приложения;
* введение системы достижений и рейтингов.

Также интересным направлением может стать разработка редактора сценариев, позволяющего игрокам создавать собственные катастрофы, карты и условия победы.

Проведённая работа имеет не только развлекательную, но и научно-прикладную ценность. Разработанная модель оценки шансов на выживание с учётом множества переменных может найти применение в психологических исследованиях, теориях принятия решений и даже в системах моделирования кризисных ситуаций. С методической точки зрения, проект демонстрирует эффективный подход к разработке сложных игровых систем с использованием современных парадигм программирования.

Таким образом, данный курсовой проект представляет собой законченное игровое приложение, сочетающее в себе: увлекательный геймплей, глубокую механику принятия решений, качественную визуальную реализацию, гибкую систему настройки параметров игры.

Проект может быть использован как для развлекательных целей, так и в образовательном процессе для развития навыков стратегического мышления и анализа сложных ситуаций.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Григорьев А.Б. О чем не пишут в книгах по Delphi [Текст]. – СПб.: БХВ-Петербург, 2010. – 576 с. – ISBN 978-5-699-40703-3.
2. Культин Н.Б. Delphi в задачах и примерах [Текст]. – СПб.: БХВ-Петербург, 2012. – 288 с. – ISBN 978-5-9775-0811-7.
3. Осипов Д.Л. Алгоритмы и структуры данных в Delphi [Текст]. – СПб.: БХВ-Петербург, 2011. – 752 с. – ISBN 978-5-9775-0659-5.
4. Фаронов В. Delphi. Программирование на языке высокого уровня [Текст]: Учебник для вузов. – СПб.: Питер, 2009. – 640 с. – ISBN 978-5-9775-0657-1.
5. Фаулер М. Предметно-ориентированные языки программирования / [Текст]. – М.: Вильямс, 2011. – 576 с. – ISBN 978-5-8459-1738-6.
6. Фленов М.Е. Библия Delphi [Текст]. 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2008. – 800 с. – ISBN 978-5-397-01304-8.
7. Чиртик А. Программирование в Delphi. Трюки и эффекты [Текст]. – СПб.: Питер, 2011. – 400 с. – ISBN 978-5-8046-0008-3.
8. Официальная документация Embarcadero Delphi [Электронный ресурс]. URL: https://docwiki.embarcadero.com/.
9. Осипов Д. Л. Базы данных и Delphi. Теория и практика. – СПб.: БХВ-Петербург, 2011. – 752 с. – ISBN 978-5-9775-0659-5.

# ПРИЛОЖЕНИЕ A

Исходный код

Unit StartUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes,

Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls,

Vcl.ExtCtrls,

Vcl.Imaging.Pngimage, Vcl.Menus, CardSystem, MainUnit;

Const

MAX\_PLAYERS = 16;

Type

TSaveFile = File Of TGameSave;

TStartMenuForm = Class(TForm)

ExitButtonLabel: TLabel;

StartMainMenu: TMainMenu;

AboutDeveloperMenuItem: TMenuItem;

RulesLabel: TLabel;

Zagolovok: TImage;

Background: TImage;

NewGameLabel: TLabel;

ConnectionLabel: TLabel;

OpenFileMenuItem: TMenuItem;

Procedure ExitButtonLabelClick(Sender: TObject);

Procedure AboutDeveloperMenuItemClick(Sender: TObject);

Procedure FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Procedure LabelMouseEnter(Sender: TObject);

Procedure LabelMouseLeave(Sender: TObject);

Function FormHelp(Command: Word; Data: THelpEventData;

Var CallHelp: Boolean): Boolean;

Procedure NewGameLabelClick(Sender: TObject);

Procedure ConnectionLabelClick(Sender: TObject);

Procedure RulesLabelClick(Sender: TObject);

Procedure OpenFileMenuItemClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

StartMenuForm: TStartMenuForm;

Implementation

{$R \*.dfm}

Uses

SeedEnterUnit, ManualUnit, DeveloperUnit, PreparationUnit, RuleUnit;

Procedure TStartMenuForm.AboutDeveloperMenuItemClick(Sender: TObject);

Begin

DeveloperForm.ShowModal;

End;

Procedure TStartMenuForm.NewGameLabelClick(Sender: TObject);

Begin

StartMenuForm.Visible := False;

PreparationForm.ShowModal;

StartMenuForm.Visible := True;

End;

Procedure TStartMenuForm.ConnectionLabelClick(Sender: TObject);

Begin

StartMenuForm.Visible := False;

SeedEnterForm.ShowModal;

StartMenuForm.Visible := True;

End;

Procedure TStartMenuForm.ExitButtonLabelClick(Sender: TObject);

Begin

StartMenuForm.Close;

End;

Procedure TStartMenuForm.FormCloseQuery(Sender: TObject; Var CanClose: Boolean);

Begin

CanClose := MessageBox(Handle, 'Вы действительно хотите выйти?',

'Вы уверены?', MB\_YESNO Or MB\_ICONQUESTION) = IDYES;

End;

Function TStartMenuForm.FormHelp(Command: Word; Data: THelpEventData;

Var CallHelp: Boolean): Boolean;

Begin

CallHelp := False;

End;

Procedure TStartMenuForm.OpenFileMenuItemClick(Sender: TObject);

Begin

GenerateGameSetup(GameSetup, 1001, 4);

StartMenuForm.Visible := False;

CurrentPlayerIndex := 0;

Try

MainForm.ShowModal;

Finally

StartMenuForm.Visible := True;

End;

End;

Procedure TStartMenuForm.RulesLabelClick(Sender: TObject);

Begin

StartMenuForm.Visible := False;

RuleForm.ShowModal;

StartMenuForm.Visible := True;

End;

Procedure TStartMenuForm.LabelMouseEnter(Sender: TObject);

Begin

With Sender As TLabel Do

Begin

Font.Size := Font.Size + 5;

Top := Top - 8;

End;

End;

Procedure TStartMenuForm.LabelMouseLeave(Sender: TObject);

Begin

With Sender As TLabel Do

Begin

Font.Size := Font.Size - 5;

Top := Top + 8;

End;

End;

End.

Unit SeedEnterUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.StdCtrls, Vcl.Menus, Clipbrd, Vcl.Imaging.Pngimage, Vcl.ExtCtrls;

Type

TSeedEnterForm = Class(TForm)

Background: TImage;

Zagolovok: TImage;

BackButtonLabel: TLabel;

NextLabel: TLabel;

FirstField: TImage;

SecondField: TImage;

ThirdField: TImage;

FourthField: TImage;

El1: TImage;

El2: TImage;

El3: TImage;

El4: TImage;

El5: TImage;

El6: TImage;

El7: TImage;

El8: TImage;

El9: TImage;

El0: TImage;

ElBack: TImage;

RulesLabel: TLabel;

FirstNum: TLabel;

SecondNum: TLabel;

ThirdNum: TLabel;

FourthNum: TLabel;

Procedure BackButtonLabelClick(Sender: TObject);

Procedure NextLabelClick(Sender: TObject);

Procedure NumberClick(Sender: TObject);

Procedure NumberDblClick(Sender: TObject);

Procedure ShowSeed(Sender: TObject; Const Seed: String);

Procedure ElBackClick(Sender: TObject);

Procedure ElBackDblClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

SeedEnterForm: TSeedEnterForm;

Seed: String;

Const

KBACKSPACE = #8;

KMINUS = #45;

KCOMMA = #44;

KDOWN = 40;

KUP = 38;

KENTER = 13;

KINSERT = 45;

Implementation

{$R \*.dfm}

Uses

ConnectUnit;

Procedure TSeedEnterForm.ShowSeed(Sender: TObject; Const Seed: String);

Var

IntSeed: Integer;

I: Integer;

Begin

FirstNum.Caption := '';

SecondNum.Caption := '';

ThirdNum.Caption := '';

FourthNum.Caption := '';

If TryStrToInt(Seed, IntSeed) Then

Begin

Case Length(Seed) Of

1:

FirstNum.Caption := Seed;

2:

Begin

FirstNum.Caption := IntToStr(IntSeed Div 10);

SecondNum.Caption := IntToStr(IntSeed Mod 10);

End;

3:

Begin

FirstNum.Caption := IntToStr(IntSeed Div 100);

SecondNum.Caption := IntToStr(IntSeed Div 10 Mod 10);

ThirdNum.Caption := IntToStr(IntSeed Mod 10);

End;

4:

Begin

FirstNum.Caption := IntToStr(IntSeed Div 1000);

SecondNum.Caption := IntToStr(IntSeed Div 100 Mod 10);

ThirdNum.Caption := IntToStr(IntSeed Div 10 Mod 10);

FourthNum.Caption := IntToStr(IntSeed Mod 10);

End;

End;

End;

End;

Procedure TSeedEnterForm.BackButtonLabelClick(Sender: TObject);

Begin

Close;

End;

Procedure TSeedEnterForm.NumberClick(Sender: TObject);

Begin

If (Sender Is TImage) And (Length(Seed) < 1) Then

Begin

Seed := Seed + Copy(TImage(Sender).Name, 3, 1);

If Seed = '0' Then

Begin

Seed := '';

End;

ShowSeed(Self, Seed);

End

Else If (Sender Is TImage) And (Length(Seed) < 4) Then

Begin

Seed := Seed + Copy(TImage(Sender).Name, 3, 1);

ShowSeed(Self, Seed);

End;

End;

Procedure TSeedEnterForm.NumberDblClick(Sender: TObject);

Begin

NumberClick(Sender);

End;

Procedure TSeedEnterForm.ElBackClick(Sender: TObject);

Begin

If Length(Seed) > 0 Then

Begin

Delete(Seed, Length(Seed), 1);

ShowSeed(Self, Seed);

End;

End;

Procedure TSeedEnterForm.ElBackDblClick(Sender: TObject);

Begin

ElBackClick(Sender);

End;

Function GetPlayerCountFromSeed(Seed: Integer): Integer;

Const

TOTAL\_SEEDS = 9000;

PLAYER\_RANGES = 13;

SEEDS\_PER\_PLAYER = TOTAL\_SEEDS Div PLAYER\_RANGES;

Begin

Result := 4 + ((Seed - 1000) Div SEEDS\_PER\_PLAYER);

If Seed >= 1000 + (SEEDS\_PER\_PLAYER \* 12) Then

Result := 16;

End;

Procedure TSeedEnterForm.NextLabelClick(Sender: TObject);

Begin

If Length(Seed) = 4 Then

Begin

Visible := False;

IntSeed := StrToInt(Seed);

ColPlayers := GetPlayerCountFromSeed(IntSeed);

Try

ConnectForm.ShowModal;

Finally

Visible := True;

End;

End;

End;

End.

Unit RuleUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Imaging.Pngimage, Vcl.ExtCtrls;

Type

TRuleForm = Class(TForm)

Rule1: TImage;

Rule2: TImage;

Rule3: TImage;

Procedure Rule1Click(Sender: TObject);

Procedure Rule2Click(Sender: TObject);

Procedure Rule3Click(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

RuleForm: TRuleForm;

Implementation

{$R \*.dfm}

Procedure TRuleForm.FormCreate(Sender: TObject);

Begin

Rule1.Visible := True;

Rule2.Visible := False;

Rule3.Visible := False;

End;

Procedure TRuleForm.Rule1Click(Sender: TObject);

Begin

Rule1.Visible := False;

Rule2.Visible := True;

End;

Procedure TRuleForm.Rule2Click(Sender: TObject);

Begin

Rule2.Visible := False;

Rule3.Visible := True;

End;

Procedure TRuleForm.Rule3Click(Sender: TObject);

Begin

Rule3.Visible := False;

Rule1.Visible := True;

End;

End.

Unit PreparationUnit;

Interface

Uses

System.DateUtils, Winapi.Windows, Winapi.Messages, System.SysUtils,

System.Variants, System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms,

Vcl.Dialogs, Vcl.StdCtrls, Vcl.Mask, Vcl.Menus, Clipbrd, MainUnit,

Vcl.Imaging.Pngimage, Vcl.ExtCtrls;

Type

TPreparationForm = Class(TForm)

Background: TImage;

PlayersCol: TImage;

BackButtonLabel: TLabel;

Left: TImage;

Right: TImage;

ColPlayersLabel: TLabel;

Zagolovok: TImage;

RulesLabel: TLabel;

NextLabel: TLabel;

Procedure BackButtonLabelClick(Sender: TObject);

Procedure LeftClick(Sender: TObject);

Procedure RightClick(Sender: TObject);

Procedure NextLabelClick(Sender: TObject);

Procedure RightDblClick(Sender: TObject);

Procedure LeftDblClick(Sender: TObject);

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

PreparationForm: TPreparationForm;

Implementation

{$R \*.dfm}

Uses

ConnectUnit;

Const

KBACKSPACE = #8;

KINSERT = 45;

MIN\_SEED = 1000;

MAX\_SEED = 9999;

PLAYER\_RANGES = 13;

MIN\_PLAYERS = 4;

MAX\_PLAYERS = 16;

Function GenerateRandomSeed(PlayerCount: Integer): Integer;

Var

SystemTime: TSystemTime;

TickCount: DWORD;

ProcessID: DWORD;

CombinedValue: Int64;

RangeSize, RangeStart, RangeEnd: Integer;

Begin

GetSystemTime(SystemTime);

TickCount := GetTickCount;

ProcessID := GetCurrentProcessId;

CombinedValue := (Int64(SystemTime.WSecond) Shl 56) Or

(Int64(SystemTime.WMilliseconds) Shl 44) Or (Int64(TickCount) Shl 32) Or

(Int64(ProcessID) Shl 16) Or (Int64(Random(65536)));

RangeSize := (MAX\_SEED - MIN\_SEED + 1) Div PLAYER\_RANGES;

RangeStart := MIN\_SEED + ((PlayerCount - MIN\_PLAYERS) Mod PLAYER\_RANGES) \*

RangeSize;

RangeEnd := RangeStart + RangeSize - 1;

If (PlayerCount - MIN\_PLAYERS) Mod PLAYER\_RANGES = PLAYER\_RANGES - 1 Then

RangeEnd := MAX\_SEED;

GenerateRandomSeed := RangeStart +

(Abs(CombinedValue) Mod (RangeEnd - RangeStart + 1));

End;

Procedure TPreparationForm.BackButtonLabelClick(Sender: TObject);

Begin

Close;

End;

Procedure TPreparationForm.LeftClick(Sender: TObject);

Begin

If StrToInt(ColPlayersLabel.Caption) = MIN\_PLAYERS Then

ColPlayersLabel.Caption := IntToStr(MAX\_PLAYERS)

Else

ColPlayersLabel.Caption :=

IntToStr(StrToInt(ColPlayersLabel.Caption) - 1);

End;

Procedure TPreparationForm.LeftDblClick(Sender: TObject);

Begin

LeftClick(Sender);

End;

Procedure TPreparationForm.NextLabelClick(Sender: TObject);

Begin

ColPlayers := StrToInt(ColPlayersLabel.Caption);

IntSeed := GenerateRandomSeed(ColPlayers);

PreparationForm.Visible := False;

Try

ConnectForm.ShowModal;

Finally

PreparationForm.Visible := True;

End;

End;

Procedure TPreparationForm.RightClick(Sender: TObject);

Begin

If StrToInt(ColPlayersLabel.Caption) = MAX\_PLAYERS Then

ColPlayersLabel.Caption := IntToStr(MIN\_PLAYERS)

Else

ColPlayersLabel.Caption :=

IntToStr(StrToInt(ColPlayersLabel.Caption) + 1);

End;

Procedure TPreparationForm.RightDblClick(Sender: TObject);

Begin

RightClick(Sender);

End;

End.

Unit MainUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.Menus, Vcl.StdCtrls, Vcl.Grids, Vcl.Imaging.Pngimage, Vcl.ExtCtrls,

CardSystem, System.ImageList, Vcl.ImgList, Vcl.VirtualImageList,

Vcl.BaseImageCollection, Vcl.ImageCollection, Vcl.VirtualImage,

FinalUnit;

Const

MAX\_PLAYERS = 16;

Type

TCardCategory = (CcProfession, CcBiology, CcHealth, CcHobby,

CcLuggage, CcFact);

TCardCategoryHelper = Record Helper For TCardCategory

Const

First = CcProfession;

Last = CcFact;

Count = Ord(Last) + 1;

End;

TPlayerCardState = Record

Revealed: Boolean;

Kicked: Boolean;

CardText: Array [TCardCategory] Of String;

End;

TPlayerCardSave = Record

Revealed: Boolean;

Kicked: Boolean;

CardText: Array [TCardCategory] Of String[255];

End;

TGameSave = Record

Players: Integer;

CurrentPlayerIndex: Integer;

ViewedPlayer: Integer;

CurrentCategory: TCardCategory;

CatastropheTitle: String[255];

CatastropheDesc: String[255];

PlayerCards: Array [1 .. MAX\_PLAYERS] Of TPlayerCardSave;

End;

TSaveFile = File Of TGameSave;

TMainForm = Class(TForm)

Background: TImage;

Left: TImage;

Right: TImage;

PlayerLabel: TLabel;

Card: TImage;

BackButtonLabel: TLabel;

ProfLabel: TLabel;

CardRightChange: TImage;

CardLeftChange: TImage;

Timer: TTimer;

ImageCollection: TImageCollection;

VirtualImageList: TVirtualImageList;

HelpButton: TImage;

RuleTable: TImage;

Rules: TImage;

RulesTimer: TTimer;

CataclysmLabel: TLabel;

CatastropheTitle: TLabel;

KickButtonCollection: TImageCollection;

KickButton: TImage;

KickButtonList: TVirtualImageList;

NextLabel: TLabel;

MainMenu: TMainMenu;

N1: TMenuItem;

SaveFileMenuItem: TMenuItem;

ExitMenuItem: TMenuItem;

OpenFileMenuItem: TMenuItem;

Procedure BackButtonLabelClick(Sender: TObject);

Procedure RightClick(Sender: TObject);

Procedure LeftClick(Sender: TObject);

Procedure CardLeftClick(Sender: TObject);

Procedure CardRightClick(Sender: TObject);

Procedure FormShow(Sender: TObject);

Procedure CardClick(Sender: TObject);

Procedure TimerTimer(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure KickButtonClick(Sender: TObject);

Procedure HelpButtonClick(Sender: TObject);

Procedure FormPaint(Sender: TObject);

Procedure RulesClick(Sender: TObject);

Procedure RulesTimerTimer(Sender: TObject);

Procedure CataclysmLabelOnImage(Image: TImage; ALabel: TLabel);

Procedure NextLabelClick(Sender: TObject);

Procedure SaveFileMenuItemClick(Sender: TObject);

Procedure OpenFileMenuItemClick(Sender: TObject);

Private Const

CARD\_BACK\_INDEX = 6;

MIN\_PLAYERS = 1;

ALPHA\_STEP = 45;

BETA\_STEP = 10;

MAX\_ALPHA = 255;

MIN\_BETA = -340;

MAX\_BETA = 0;

DEFAULT\_IMAGE\_WIDTH = 389;

DEFAULT\_IMAGE\_HEIGHT = 549;

DEFAULT\_IMAGE\_LEFT = 332;

Private

FCurrentIndex: Integer;

FTargetIndex: Integer;

FAlpha: Integer;

FBeta: Integer;

IsBack: Boolean;

FSourceBmp: TBitmap;

FTargetBmp: TBitmap;

FIsDarkened: Boolean;

Procedure UpdatePlayerNumber(IsIncrease: Boolean);

Function GetCardType: TCardType;

Procedure PrepareBitmaps;

Procedure CleanupBitmap(Bmp: TBitmap);

Function GetPlayerIndex: Integer;

Function IsAnimationRunning: Boolean;

Procedure StartAnimation(TargetIndex: Integer);

Procedure SetCategory(Category: TCardCategory);

Procedure CycleCategory(IsNext: Boolean);

Procedure ToggleCardReveal;

Procedure KickCardReveal;

Function GetCategoryIndex(Category: TCardCategory): Integer;

Procedure StartDarkenAnimation(Image: TImage; NewImageIndex: Integer);

Function HasHeterosexualPair: Boolean;

Public

FViewedPlayer: Integer;

FCurrentCategory: TCardCategory;

FPlayersCards: Array Of TPlayerCardState;

Function GetCardTypeFromCategory(Category: TCardCategory): TCardType;

Procedure UpdateKickButton;

Procedure InitializePlayersCards;

Procedure ShowCurrentCard;

End;

Var

MainForm: TMainForm;

GameSetup: TGameSetup;

CurrentPlayerIndex: Integer;

Implementation

{$R \*.dfm}

{ TMainForm }

Procedure TMainForm.BackButtonLabelClick(Sender: TObject);

Begin

If MessageBox(Handle, 'Вы действительно хотите выйти?', 'Вы уверены?',

MB\_YESNO Or MB\_ICONQUESTION) = IDYES Then

Begin

Close;

End;

End;

Function TMainForm.GetCardTypeFromCategory(Category: TCardCategory): TCardType;

Begin

Case Category Of

CcProfession:

GetCardTypeFromCategory := CtProfession;

CcBiology:

GetCardTypeFromCategory := CtBiology;

CcHealth:

GetCardTypeFromCategory := CtHealth;

CcHobby:

GetCardTypeFromCategory := CtHobby;

CcLuggage:

GetCardTypeFromCategory := CtLuggage;

CcFact:

GetCardTypeFromCategory := CtFacts;

Else

GetCardTypeFromCategory := CtProfession;

End;

End;

Procedure TMainForm.RulesClick(Sender: TObject);

Begin

FIsDarkened := Not FIsDarkened;

If FIsDarkened Then

Begin

CardLeftChange.OnClick := Nil;

CardRightChange.OnClick := Nil;

Card.OnClick := Nil;

Left.OnClick := Nil;

Left.OnDblClick := Nil;

Right.OnClick := Nil;

Right.OnDblClick := Nil;

KickButton.OnClick := Nil;

BackButtonLabel.OnClick := Nil;

HelpButton.OnClick := Nil;

IsBack := False;

RulesTimer.Enabled := True;

End

Else

Begin

Repaint;

CardLeftChange.OnClick := CardLeftClick;

CardRightChange.OnClick := CardRightClick;

Card.OnClick := CardClick;

Left.OnClick := LeftClick;

Left.OnDblClick := LeftClick;

Right.OnClick := RightClick;

Right.OnDblClick := RightClick;

KickButton.OnClick := KickButtonClick;

BackButtonLabel.OnClick := BackButtonLabelClick;

HelpButton.OnClick := HelpButtonClick;

IsBack := True;

RulesTimer.Enabled := True;

End;

End;

Procedure TMainForm.InitializePlayersCards;

Var

I: Integer;

Cat: TCardCategory;

Begin

SetLength(FPlayersCards, GameSetup.Players);

For I := 0 To GameSetup.Players - 1 Do

Begin

FPlayersCards[I].Revealed := False;

FPlayersCards[I].Kicked := False;

For Cat := TCardCategory.First To TCardCategory.Last Do

Begin

FPlayersCards[I].CardText[Cat] := '';

End;

End;

If GameSetup.Players > 0 Then

Begin

FPlayersCards[CurrentPlayerIndex].Revealed := True;

For Cat := TCardCategory.First To TCardCategory.Last Do

Begin

FPlayersCards[CurrentPlayerIndex].CardText[Cat] :=

GameSetup.Cards[GetCardTypeFromCategory(Cat)

][CurrentPlayerIndex].Description;

End;

End;

CataclysmLabel.Caption := GameSetup.Catastrophe.Description;

CataclysmLabelOnImage(Rules, CataclysmLabel);

CatastropheTitle.Caption := GameSetup.Catastrophe.Title;

CatastropheTitle.Parent := Rules.Parent;

CatastropheTitle.BringToFront;

CatastropheTitle.Left := Rules.Left + 30;

CatastropheTitle.Top := Rules.Top + 480;

CatastropheTitle.Width := Rules.Width - 60;

CatastropheTitle.Height := Rules.Height - 30;

FCurrentCategory := TCardCategory.First;

End;

Function TMainForm.GetCategoryIndex(Category: TCardCategory): Integer;

Begin

GetCategoryIndex := Ord(Category);

End;

Procedure TMainForm.StartDarkenAnimation(Image: TImage; NewImageIndex: Integer);

Var

I: Integer;

DarkBmp: TBitmap;

OriginalBmp: TBitmap;

BlendFunc: TBlendFunction;

Begin

OriginalBmp := TBitmap.Create;

DarkBmp := TBitmap.Create;

Try

OriginalBmp.Assign(Image.Picture.Graphic);

OriginalBmp.PixelFormat := Pf32bit;

OriginalBmp.AlphaFormat := AfPremultiplied;

DarkBmp.SetSize(OriginalBmp.Width, OriginalBmp.Height);

DarkBmp.PixelFormat := Pf32bit;

DarkBmp.AlphaFormat := AfPremultiplied;

BlendFunc.BlendOp := AC\_SRC\_OVER;

BlendFunc.BlendFlags := 0;

BlendFunc.AlphaFormat := AC\_SRC\_ALPHA;

For I := 1 To 5 Do

Begin

DarkBmp.Canvas.Brush.Color := ClBlack;

DarkBmp.Canvas.FillRect(Rect(0, 0, DarkBmp.Width, DarkBmp.Height));

BlendFunc.SourceConstantAlpha := I \* 50;

Winapi.Windows.AlphaBlend(DarkBmp.Canvas.Handle, 0, 0,

DarkBmp.Width, DarkBmp.Height, OriginalBmp.Canvas.Handle, 0, 0,

OriginalBmp.Width, OriginalBmp.Height, BlendFunc);

Image.Picture.Assign(DarkBmp);

Image.Refresh;

End;

KickButtonList.GetBitmap(1, Image.Picture.Bitmap);

Image.Refresh;

Finally

OriginalBmp.Free;

DarkBmp.Free;

End;

End;

Procedure TMainForm.KickButtonClick(Sender: TObject);

Var

DarkBmp: TBitmap;

BlendFunc: TBlendFunction;

OriginalBmp: TBitmap;

PlayerIndex: Integer;

Begin

PlayerIndex := GetPlayerIndex;

If MessageBox(Handle, 'Вы действительно хотите выгнать игрока?',

'Вы уверены?', MB\_YESNO Or MB\_ICONQUESTION) = IDYES Then

Begin

If Sender Is TImage Then

Begin

FPlayersCards[PlayerIndex].Kicked := True;

OriginalBmp := TBitmap.Create;

DarkBmp := TBitmap.Create;

Try

OriginalBmp.Assign(TImage(Sender).Picture.Graphic);

OriginalBmp.PixelFormat := Pf32bit;

OriginalBmp.AlphaFormat := AfPremultiplied;

DarkBmp.SetSize(OriginalBmp.Width, OriginalBmp.Height);

DarkBmp.PixelFormat := Pf32bit;

DarkBmp.AlphaFormat := AfPremultiplied;

DarkBmp.Canvas.Brush.Color := ClBlack;

DarkBmp.Canvas.FillRect(Rect(0, 0, DarkBmp.Width,

DarkBmp.Height));

BlendFunc.BlendOp := AC\_SRC\_OVER;

BlendFunc.BlendFlags := 0;

BlendFunc.SourceConstantAlpha := 128;

BlendFunc.AlphaFormat := AC\_SRC\_ALPHA;

Winapi.Windows.AlphaBlend(DarkBmp.Canvas.Handle, 0, 0,

DarkBmp.Width, DarkBmp.Height, OriginalBmp.Canvas.Handle, 0,

0, OriginalBmp.Width, OriginalBmp.Height, BlendFunc);

TImage(Sender).Picture.Assign(DarkBmp);

TImage(Sender).Refresh;

UpdateKickButton;

KickCardReveal;

StartDarkenAnimation(KickButton, 0);

Finally

OriginalBmp.Free;

DarkBmp.Free;

End;

End;

End;

End;

Procedure TMainForm.UpdatePlayerNumber(IsIncrease: Boolean);

Begin

If IsIncrease Then

FViewedPlayer := (FViewedPlayer Mod GameSetup.Players) + 1

Else

FViewedPlayer := ((FViewedPlayer + GameSetup.Players - 2)

Mod GameSetup.Players) + 1;

FCurrentCategory := TCardCategory.First;

PlayerLabel.Caption := 'ИГРОК ' + IntToStr(FViewedPlayer);

ShowCurrentCard;

End;

Procedure TMainForm.UpdateKickButton;

Var

PlayerIndex: Integer;

ActivePlayers, TotalPlayers: Integer;

I: Integer;

Begin

PlayerIndex := GetPlayerIndex;

KickButton.Transparent := True;

If FPlayersCards[PlayerIndex].Kicked Then

KickButtonList.GetBitmap(1, KickButton.Picture.Bitmap)

Else

KickButtonList.GetBitmap(0, KickButton.Picture.Bitmap);

KickButton.Picture.Bitmap.AlphaFormat := AfPremultiplied;

KickButton.Picture.Bitmap.PixelFormat := Pf32bit;

KickButton.Invalidate;

TotalPlayers := GameSetup.Players;

ActivePlayers := 0;

For I := 0 To TotalPlayers - 1 Do

Begin

If Not FPlayersCards[I].Kicked Then

Inc(ActivePlayers);

End;

NextLabel.Visible := (ActivePlayers <= (TotalPlayers Div 2)) And

(TotalPlayers > 1);

KickButton.Refresh;

End;

Procedure TMainForm.ShowCurrentCard;

Var

PlayerIndex: Integer;

CardTextEmpty: Boolean;

BlendFunc: TBlendFunction;

TempBmp: TBitmap;

Begin

PlayerIndex := GetPlayerIndex;

CardTextEmpty := (FPlayersCards[PlayerIndex].CardText

[FCurrentCategory] = '');

If (FCurrentCategory = CcProfession) And Not FPlayersCards[PlayerIndex]

.Revealed And Not Timer.Enabled Then

Begin

FTargetIndex := GetCategoryIndex(FCurrentCategory);

PrepareBitmaps;

StartAnimation(CARD\_BACK\_INDEX);

ProfLabel.Caption := '';

End

Else If (Not FPlayersCards[PlayerIndex].Revealed Or CardTextEmpty) And

Not Timer.Enabled Then

Begin

FTargetIndex := GetCategoryIndex(FCurrentCategory);

PrepareBitmaps;

StartAnimation(CARD\_BACK\_INDEX);

ProfLabel.Caption := '';

End

Else If (FCurrentCategory = CcProfession) And Not FPlayersCards[PlayerIndex]

.Revealed And Timer.Enabled Then

Begin

TempBmp := TBitmap.Create;

Try

TempBmp.SetSize(FSourceBmp.Width, FSourceBmp.Height);

TempBmp.PixelFormat := Pf32bit;

TempBmp.AlphaFormat := AfPremultiplied;

TempBmp.Canvas.Brush.Color := ClBlack;

TempBmp.Canvas.FillRect(Rect(0, 0, TempBmp.Width, TempBmp.Height));

BlendFunc.BlendOp := AC\_SRC\_OVER;

BlendFunc.BlendFlags := 0;

BlendFunc.AlphaFormat := AC\_SRC\_ALPHA;

BlendFunc.SourceConstantAlpha := MAX\_ALPHA - FAlpha;

Winapi.Windows.AlphaBlend(TempBmp.Canvas.Handle, 0, 0,

TempBmp.Width, TempBmp.Height, FSourceBmp.Canvas.Handle, 0, 0,

FSourceBmp.Width, FSourceBmp.Height, BlendFunc);

BlendFunc.SourceConstantAlpha := FAlpha;

Winapi.Windows.AlphaBlend(TempBmp.Canvas.Handle, 0, 0,

TempBmp.Width, TempBmp.Height, FTargetBmp.Canvas.Handle, 0, 0,

FTargetBmp.Width, FTargetBmp.Height, BlendFunc);

Card.Picture.Bitmap.Assign(TempBmp);

Card.Invalidate;

Timer.Enabled := False;

FCurrentIndex := FTargetIndex;

VirtualImageList.GetBitmap(FCurrentIndex, Card.Picture.Bitmap);

CleanupBitmap(Card.Picture.Bitmap);

Card.Width := DEFAULT\_IMAGE\_WIDTH;

Card.Left := DEFAULT\_IMAGE\_LEFT;

ProfLabel.Visible := (FCurrentIndex <> CARD\_BACK\_INDEX) And

FPlayersCards[GetPlayerIndex].Revealed And

(FPlayersCards[GetPlayerIndex].CardText

[FCurrentCategory] <> '');

Finally

TempBmp.Free;

End;

VirtualImageList.GetBitmap(FCurrentIndex, FSourceBmp);

VirtualImageList.GetBitmap(FTargetIndex, FTargetBmp);

StartAnimation(CARD\_BACK\_INDEX);

ProfLabel.Caption := '';

End

Else

Begin

StartAnimation(GetCategoryIndex(FCurrentCategory));

ProfLabel.Caption := FPlayersCards[PlayerIndex].CardText

[FCurrentCategory];

End;

UpdateKickButton

End;

Procedure TMainForm.PrepareBitmaps;

Begin

VirtualImageList.GetBitmap(FCurrentIndex, FSourceBmp);

VirtualImageList.GetBitmap(FTargetIndex, FTargetBmp);

CleanupBitmap(FSourceBmp);

CleanupBitmap(FTargetBmp);

End;

Procedure TMainForm.CleanupBitmap(Bmp: TBitmap);

Var

X, Y: Integer;

P: PRGBQuad;

Begin

Bmp.PixelFormat := Pf32bit;

Bmp.AlphaFormat := AfPremultiplied;

For Y := 0 To Bmp.Height - 1 Do

Begin

P := Bmp.ScanLine[Y];

For X := 0 To Bmp.Width - 1 Do

Begin

If P.RgbReserved = 0 Then

Begin

P.RgbBlue := 0;

P.RgbGreen := 0;

P.RgbRed := 0;

End;

Inc(P);

End;

End;

End;

Function Min(A, B: Integer): Integer;

Begin

If A > B Then

Min := B

Else If A < B Then

Min := A

Else

Min := A;

End;

Function Max(A, B: Integer): Integer;

Begin

If A > B Then

Max := A

Else If A < B Then

Max := B

Else

Max := A;

End;

Procedure TMainForm.RulesTimerTimer(Sender: TObject);

Begin

If IsBack Then

Begin

Rules.Left := Rules.Left - BETA\_STEP;

CataclysmLabelOnImage(Rules, CataclysmLabel);

CatastropheTitle.Parent := Rules.Parent;

CatastropheTitle.BringToFront;

CatastropheTitle.Left := Rules.Left + 30;

CatastropheTitle.Top := Rules.Top + 460;

CatastropheTitle.Width := Rules.Width - 60;

CatastropheTitle.Height := Rules.Height - 30;

If Rules.Left <= MIN\_BETA Then

Begin

RulesTimer.Enabled := False;

End;

End

Else

Begin

Rules.Left := Rules.Left + BETA\_STEP;

CataclysmLabelOnImage(Rules, CataclysmLabel);

CatastropheTitle.Parent := Rules.Parent;

CatastropheTitle.BringToFront;

CatastropheTitle.Left := Rules.Left + 30;

CatastropheTitle.Top := Rules.Top + 460;

CatastropheTitle.Width := Rules.Width - 60;

CatastropheTitle.Height := Rules.Height - 30;

If Rules.Left >= MAX\_BETA Then

Begin

RulesTimer.Enabled := False;

End;

End;

End;

Procedure TMainForm.TimerTimer(Sender: TObject);

Var

BlendFunc: TBlendFunction;

TempBmp: TBitmap;

Begin

FAlpha := Min(FAlpha + ALPHA\_STEP, MAX\_ALPHA);

If Card.Width < DEFAULT\_IMAGE\_WIDTH Then

Begin

Card.Width := Card.Width + (DEFAULT\_IMAGE\_WIDTH - Card.Width) Div 4 + 1;

Card.Left := Card.Left - (DEFAULT\_IMAGE\_WIDTH - Card.Width) Div 8;

End;

TempBmp := TBitmap.Create;

Try

TempBmp.SetSize(FSourceBmp.Width, FSourceBmp.Height);

TempBmp.PixelFormat := Pf32bit;

TempBmp.AlphaFormat := AfPremultiplied;

TempBmp.Canvas.Brush.Color := ClBlack;

TempBmp.Canvas.FillRect(Rect(0, 0, TempBmp.Width, TempBmp.Height));

BlendFunc.BlendOp := AC\_SRC\_OVER;

BlendFunc.BlendFlags := 0;

BlendFunc.AlphaFormat := AC\_SRC\_ALPHA;

BlendFunc.SourceConstantAlpha := MAX\_ALPHA - FAlpha;

Winapi.Windows.AlphaBlend(TempBmp.Canvas.Handle, 0, 0, TempBmp.Width,

TempBmp.Height, FSourceBmp.Canvas.Handle, 0, 0, FSourceBmp.Width,

FSourceBmp.Height, BlendFunc);

BlendFunc.SourceConstantAlpha := FAlpha;

Winapi.Windows.AlphaBlend(TempBmp.Canvas.Handle, 0, 0, TempBmp.Width,

TempBmp.Height, FTargetBmp.Canvas.Handle, 0, 0, FTargetBmp.Width,

FTargetBmp.Height, BlendFunc);

Card.Picture.Bitmap.Assign(TempBmp);

Card.Invalidate;

If FAlpha >= MAX\_ALPHA Then

Begin

Timer.Enabled := False;

FCurrentIndex := FTargetIndex;

VirtualImageList.GetBitmap(FCurrentIndex, Card.Picture.Bitmap);

CleanupBitmap(Card.Picture.Bitmap);

Card.Width := DEFAULT\_IMAGE\_WIDTH;

Card.Left := DEFAULT\_IMAGE\_LEFT;

ProfLabel.Visible := (FCurrentIndex <> CARD\_BACK\_INDEX) And

FPlayersCards[GetPlayerIndex].Revealed And

(FPlayersCards[GetPlayerIndex].CardText

[FCurrentCategory] <> '');

End;

Finally

TempBmp.Free;

End;

End;

Function TMainForm.GetCardType: TCardType;

Begin

GetCardType := GetCardTypeFromCategory(FCurrentCategory);

End;

Procedure TMainForm.FormCreate(Sender: TObject);

Begin

NextLabel.Visible := False;

IsBack := False;

RulesTimer.Enabled := False;

RuleTable.Visible := False;

FIsDarkened := False;

Timer.Enabled := False;

VirtualImageList.Scaled := True;

VirtualImageList.DrawingStyle := DsTransparent;

VirtualImageList.BkColor := ClNone;

VirtualImageList.Width := DEFAULT\_IMAGE\_WIDTH;

VirtualImageList.Height := DEFAULT\_IMAGE\_HEIGHT;

KickButtonList.Scaled := True;

KickButtonList.DrawingStyle := DsTransparent;

KickButtonList.BkColor := ClNone;

KickButtonList.Width := DEFAULT\_IMAGE\_WIDTH;

KickButtonList.Height := DEFAULT\_IMAGE\_HEIGHT;

FCurrentIndex := 0;

FTargetIndex := 0;

FAlpha := MAX\_ALPHA;

FSourceBmp := TBitmap.Create;

FSourceBmp.PixelFormat := Pf32bit;

FSourceBmp.AlphaFormat := AfPremultiplied;

FTargetBmp := TBitmap.Create;

FTargetBmp.PixelFormat := Pf32bit;

FTargetBmp.AlphaFormat := AfPremultiplied;

VirtualImageList.GetBitmap(FCurrentIndex, Card.Picture.Bitmap);

KickButtonList.GetBitmap(0, KickButton.Picture.Bitmap);

CleanupBitmap(Card.Picture.Bitmap);

CleanupBitmap(KickButton.Picture.Bitmap);

Card.Transparent := True;

KickButton.Transparent := True;

KickButton.ComponentIndex := 0;

End;

Procedure TMainForm.FormPaint(Sender: TObject);

Var

BlendFunc: TBlendFunction;

Rect: TRect;

DarkBitmap: TBitmap;

Begin

If FIsDarkened Then

Begin

Rect := ClientRect;

DarkBitmap := TBitmap.Create;

Try

DarkBitmap.Width := Rect.Width;

DarkBitmap.Height := Rect.Height;

DarkBitmap.Canvas.Brush.Color := ClBlack;

DarkBitmap.Canvas.FillRect(Rect);

BlendFunc.BlendOp := AC\_SRC\_OVER;

BlendFunc.BlendFlags := 0;

BlendFunc.SourceConstantAlpha := 128;

BlendFunc.AlphaFormat := 0;

Winapi.Windows.AlphaBlend(Canvas.Handle, Rect.Left, Rect.Top,

Rect.Width, Rect.Height, DarkBitmap.Canvas.Handle, 0, 0,

DarkBitmap.Width, DarkBitmap.Height, BlendFunc);

Finally

DarkBitmap.Free;

End;

End;

End;

Procedure TMainForm.FormShow(Sender: TObject);

Begin

FViewedPlayer := 1;

FCurrentCategory := TCardCategory.First;

PlayerLabel.Caption := 'ИГРОК 1';

Visible := True;

InitializePlayersCards;

Timer.Enabled := True;

ShowCurrentCard;

End;

Procedure TMainForm.LeftClick(Sender: TObject);

Begin

UpdatePlayerNumber(False);

End;

Function TMainForm.HasHeterosexualPair: Boolean;

Var

I: Integer;

HasMan, HasWoman: Boolean;

BiologyText: String;

Begin

HasMan := False;

HasWoman := False;

For I := 0 To High(FPlayersCards) Do

Begin

If FPlayersCards[I].Kicked Then

Continue;

BiologyText := FPlayersCards[I].CardText[CcBiology];

If (Pos('МУЖЧИНА', BiologyText) > 0) And

(Pos('ГОМОСЕКСУАЛЕН', BiologyText) = 0) Then

HasMan := True;

If (Pos('ЖЕНЩИНА', BiologyText) > 0) And

(Pos('ГОМОСЕКСУАЛЬНА', BiologyText) = 0) Then

HasWoman := True;

If (Pos('БИСЕКСУАЛЕН', BiologyText) > 0) Or

(Pos('БИСЕКСУАЛЬНА', BiologyText) > 0) Then

Begin

If Pos('МУЖЧИНА', BiologyText) > 0 Then

HasMan := True

Else If Pos('ЖЕНЩИНА', BiologyText) > 0 Then

HasWoman := True;

End;

If (Pos('ГЕРМАФРОДИТ', BiologyText) > 0) Then

Begin

HasMan := True;

HasWoman := True;

End;

End;

Result := HasMan And HasWoman;

End;

Procedure TMainForm.NextLabelClick(Sender: TObject);

Begin

FinalForm.HasPair := HasHeterosexualPair;

FinalForm.FGameSetup := GameSetup;

FinalForm.ShowModal;

Close;

End;

Procedure TMainForm.OpenFileMenuItemClick(Sender: TObject);

Var

OpenDialog: TOpenDialog;

F: TSaveFile;

GameSave: TGameSave;

I: Integer;

Cat: TCardCategory;

Begin

OpenDialog := TOpenDialog.Create(Nil);

Try

OpenDialog.Filter := 'Сохранение игры (\*.sav)|\*.sav';

OpenDialog.DefaultExt := 'sav';

OpenDialog.Title := 'Загрузить игру';

If OpenDialog.Execute Then

Begin

AssignFile(F, OpenDialog.FileName);

Reset(F);

Try

Read(F, GameSave);

GameSetup.Players := GameSave.Players;

CurrentPlayerIndex := GameSave.CurrentPlayerIndex;

FViewedPlayer := GameSave.ViewedPlayer;

FCurrentCategory := GameSave.CurrentCategory;

GameSetup.Catastrophe.Title := GameSave.CatastropheTitle;

GameSetup.Catastrophe.Description := GameSave.CatastropheDesc;

SetLength(FPlayersCards, GameSave.Players);

For I := 1 To GameSave.Players Do

Begin

FPlayersCards[I - 1].Revealed := GameSave.PlayerCards

[I].Revealed;

FPlayersCards[I - 1].Kicked := GameSave.PlayerCards

[I].Kicked;

For Cat := TCardCategory.First To TCardCategory.Last Do

FPlayersCards[I - 1].CardText[Cat] :=

GameSave.PlayerCards[I].CardText[Cat];

End;

PlayerLabel.Caption := 'ИГРОК ' + IntToStr(FViewedPlayer);

CataclysmLabel.Caption := GameSetup.Catastrophe.Description;

CataclysmLabelOnImage(Rules, CataclysmLabel);

CatastropheTitle.Caption := GameSetup.Catastrophe.Title;

ShowCurrentCard;

UpdateKickButton;

Finally

CloseFile(F);

End;

End;

Finally

OpenDialog.Free;

End;

End;

Procedure TMainForm.RightClick(Sender: TObject);

Begin

UpdatePlayerNumber(True);

End;

Procedure TMainForm.CardClick(Sender: TObject);

Begin

ToggleCardReveal;

End;

Procedure TMainForm.CardLeftClick(Sender: TObject);

Begin

CycleCategory(False);

End;

Procedure TMainForm.CardRightClick(Sender: TObject);

Begin

CycleCategory(True);

End;

Procedure AdjustLabelFontSize(ALabel: TLabel);

Var

TextRect: TRect;

Flags: Cardinal;

Begin

TextRect := Rect(0, 0, ALabel.Width, ALabel.Height);

Flags := DT\_CENTER Or DT\_WORDBREAK Or DT\_CALCRECT Or DT\_NOPREFIX;

While True Do

Begin

DrawText(ALabel.Canvas.Handle, PChar(ALabel.Caption),

Length(ALabel.Caption), TextRect, Flags);

If (TextRect.Bottom <= ALabel.Height) Or (ALabel.Font.Size <= 6) Then

Break;

ALabel.Font.Size := ALabel.Font.Size - 1;

TextRect := Rect(0, 0, ALabel.Width, ALabel.Height);

End;

End;

Procedure TMainForm.CataclysmLabelOnImage(Image: TImage; ALabel: TLabel);

Var

TextRect: TRect;

Flags: Cardinal;

Begin

ALabel.Parent := Image.Parent;

ALabel.BringToFront;

ALabel.Left := Image.Left + 30;

ALabel.Top := Image.Top + 15;

ALabel.Width := Image.Width - 60;

ALabel.Height := Image.Height - 30;

AdjustLabelFontSize(ALabel);

End;

Function TMainForm.GetPlayerIndex: Integer;

Begin

GetPlayerIndex := FViewedPlayer - 1;

End;

Procedure TMainForm.HelpButtonClick(Sender: TObject);

Begin

FIsDarkened := Not FIsDarkened;

If FIsDarkened Then

Begin

CardLeftChange.OnClick := Nil;

CardRightChange.OnClick := Nil;

Card.OnClick := Nil;

Left.OnClick := Nil;

Left.OnDblClick := Nil;

Right.OnClick := Nil;

Right.OnDblClick := Nil;

KickButton.OnClick := Nil;

BackButtonLabel.OnClick := Nil;

FormPaint(MainForm);

RuleTable.Visible := True;

HelpButton.Enabled := False;

HelpButton.Enabled := True;

Rules.OnClick := NIL;

End

Else

Begin

Repaint;

CardLeftChange.OnClick := CardLeftClick;

CardRightChange.OnClick := CardRightClick;

Card.OnClick := CardClick;

Left.OnClick := LeftClick;

Left.OnDblClick := LeftClick;

Right.OnClick := RightClick;

Right.OnDblClick := RightClick;

KickButton.OnClick := KickButtonClick;

BackButtonLabel.OnClick := BackButtonLabelClick;

RuleTable.Visible := False;

Rules.OnClick := RulesClick;

End;

End;

Function TMainForm.IsAnimationRunning: Boolean;

Begin

IsAnimationRunning := Timer.Enabled;

End;

Procedure TMainForm.StartAnimation(TargetIndex: Integer);

Begin

If IsAnimationRunning Then

Exit;

FTargetIndex := TargetIndex;

PrepareBitmaps;

FAlpha := 0;

Timer.Enabled := True;

Card.Width := Card.Width Div 2;

Card.Left := Card.Left + Card.Width Div 2;

End;

Procedure TMainForm.SaveFileMenuItemClick(Sender: TObject);

Var

SaveDialog: TSaveDialog;

F: TSaveFile;

GameSave: TGameSave;

I: Integer;

Cat: TCardCategory;

Begin

SaveDialog := TSaveDialog.Create(Nil);

Try

SaveDialog.Filter := 'Сохранение игры (\*.sav)|\*.sav';

SaveDialog.DefaultExt := 'sav';

SaveDialog.Title := 'Сохранить игру';

If SaveDialog.Execute Then

Begin

If GameSetup.Players > MAX\_PLAYERS Then

Begin

ShowMessage('Ошибка: слишком много игроков для сохранения');

Exit;

End;

AssignFile(F, SaveDialog.FileName);

Rewrite(F);

Try

GameSave.Players := GameSetup.Players;

GameSave.CurrentPlayerIndex := CurrentPlayerIndex;

GameSave.ViewedPlayer := FViewedPlayer;

GameSave.CurrentCategory := FCurrentCategory;

GameSave.CatastropheTitle := GameSetup.Catastrophe.Title;

GameSave.CatastropheDesc := GameSetup.Catastrophe.Description;

For I := 1 To GameSetup.Players Do

Begin

GameSave.PlayerCards[I].Revealed :=

FPlayersCards[I - 1].Revealed;

GameSave.PlayerCards[I].Kicked :=

FPlayersCards[I - 1].Kicked;

For Cat := TCardCategory.First To TCardCategory.Last Do

GameSave.PlayerCards[I].CardText[Cat] :=

FPlayersCards[I - 1].CardText[Cat];

End;

Write(F, GameSave);

Finally

CloseFile(F);

End;

ShowMessage('Игра успешно сохранена!');

End;

Finally

SaveDialog.Free;

End;

End;

Procedure TMainForm.SetCategory(Category: TCardCategory);

Begin

FCurrentCategory := Category;

End;

Procedure TMainForm.CycleCategory(IsNext: Boolean);

Var

NewCategory: TCardCategory;

Begin

ProfLabel.Visible := False;

If IsNext Then

Begin

If FCurrentCategory = TCardCategory.Last Then

NewCategory := TCardCategory.First

Else

NewCategory := Succ(FCurrentCategory);

End

Else

Begin

If FCurrentCategory = TCardCategory.First Then

NewCategory := TCardCategory.Last

Else

NewCategory := Pred(FCurrentCategory);

End;

SetCategory(NewCategory);

ShowCurrentCard;

End;

Procedure TMainForm.ToggleCardReveal;

Var

PlayerIndex: Integer;

Begin

ProfLabel.Visible := False;

PlayerIndex := GetPlayerIndex;

FPlayersCards[PlayerIndex].Revealed := True;

FPlayersCards[PlayerIndex].CardText[FCurrentCategory] :=

GameSetup.Cards[GetCardTypeFromCategory(FCurrentCategory)][PlayerIndex]

.Description;

ShowCurrentCard;

End;

Procedure TMainForm.KickCardReveal;

Var

PlayerIndex: Integer;

Begin

ProfLabel.Visible := False;

PlayerIndex := GetPlayerIndex;

FPlayersCards[PlayerIndex].Revealed := True;

For Var Cat := CcProfession To CcFact Do

FPlayersCards[PlayerIndex].CardText[Cat] :=

GameSetup.Cards[GetCardTypeFromCategory(Cat)][PlayerIndex]

.Description;

ShowCurrentCard;

End;

End.

Unit FinalUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Imaging.Pngimage, Vcl.ExtCtrls,

Vcl.StdCtrls, Vcl.ComCtrls, CardSystem, Vcl.Imaging.Jpeg;

Type

PBunkerSituationNode = ^TBunkerSituationNode;

TBunkerSituationNode = Record

Situation: TBunkerSituation;

Next: PBunkerSituationNode;

End;

TBunkerSituationList = Record

First: PBunkerSituationNode;

Last: PBunkerSituationNode;

Count: Integer;

End;

TFinalForm = Class(TForm)

Background: TImage;

BackButtonLabel: TLabel;

Zagolovok: TImage;

NoButton: TImage;

YesButton: TImage;

QuestionImage: TImage;

AnimationTimer: TTimer;

BunkerSituationImage: TImage;

SituationTitle: TLabel;

BunkerSituation: TLabel;

InfoLabel: TLabel;

Win: TImage;

Lose: TImage;

Procedure BackButtonLabelClick(Sender: TObject);

Procedure FormShow(Sender: TObject);

Procedure FormCreate(Sender: TObject);

Procedure FormDestroy(Sender: TObject);

Procedure AnimationTimerTimer(Sender: TObject);

Procedure NoButtonClick(Sender: TObject);

Procedure YesButtonClick(Sender: TObject);

Private

FadeAlpha: Integer;

SituationIndex: Integer;

Chance: Integer;

AnimationStep: Integer;

BlinkCounter: Integer;

Percent: Integer;

CurrentTextPos: Integer;

FullMessage: String;

FSituations: TBunkerSituationList; // Наш список ситуаций

CurrentSituation: PBunkerSituationNode; // Текущая ситуация

Procedure ShowNextImage;

Procedure FadeInImage(Img: TImage);

Procedure BlinkImage(Img: TImage);

Procedure InitializeSituation(SituationNode: PBunkerSituationNode);

Function IsAllSituation: Boolean;

Procedure CountTheChance;

Procedure LoadSituationsFromGameSetup;

Procedure ClearSituations;

Public

FGameSetup: TGameSetup;

HasPair: Boolean;

End;

Var

FinalForm: TFinalForm;

Implementation

{$R \*.dfm}

Procedure RestartApplication;

Var

AppPath: Array [0 .. MAX\_PATH] Of Char;

ProcessInfo: TProcessInformation;

StartupInfo: TStartupInfo;

Begin

GetModuleFileName(0, AppPath, SizeOf(AppPath));

FillChar(StartupInfo, SizeOf(StartupInfo), 0);

StartupInfo.Cb := SizeOf(StartupInfo);

If CreateProcess(Nil, AppPath, Nil, Nil, False, 0, Nil, Nil, StartupInfo,

ProcessInfo) Then

Begin

CloseHandle(ProcessInfo.HThread);

CloseHandle(ProcessInfo.HProcess);

Application.Terminate;

End;

End;

Procedure TFinalForm.ClearSituations;

Var

Current, Next: PBunkerSituationNode;

Begin

Current := FSituations.First;

While Current <> Nil Do

Begin

Next := Current^.Next;

Dispose(Current);

Current := Next;

End;

FSituations.First := Nil;

FSituations.Last := Nil;

FSituations.Count := 0;

End;

Procedure TFinalForm.LoadSituationsFromGameSetup;

Var

I: Integer;

NewNode: PBunkerSituationNode;

Begin

ClearSituations;

For I := 0 To High(FGameSetup.BunkerSituations) Do

Begin

New(NewNode);

NewNode^.Situation := FGameSetup.BunkerSituations[I];

NewNode^.Next := Nil;

If FSituations.First = Nil Then

Begin

FSituations.First := NewNode;

FSituations.Last := NewNode;

End

Else

Begin

FSituations.Last^.Next := NewNode;

FSituations.Last := NewNode;

End;

Inc(FSituations.Count);

End;

End;

Procedure TFinalForm.BackButtonLabelClick(Sender: TObject);

Begin

If MessageBox(Handle, 'Вы действительно хотите выйти?', 'Подтверждение',

MB\_YESNO Or MB\_ICONQUESTION) = IDYES Then

Begin

RestartApplication;

End;

End;

Procedure TFinalForm.FormCreate(Sender: TObject);

Begin

FSituations.First := Nil;

FSituations.Last := Nil;

FSituations.Count := 0;

CurrentSituation := Nil;

Win.Visible := False;

Lose.Visible := False;

InfoLabel.Visible := False;

CurrentTextPos := 0;

FullMessage := '';

SituationIndex := 0;

BunkerSituation.Visible := False;

SituationTitle.Visible := False;

BunkerSituationImage.Visible := False;

QuestionImage.Visible := False;

YesButton.Visible := False;

YesButton.Top := 544;

YesButton.Left := 688;

NoButton.Visible := False;

NoButton.Top := 544;

NoButton.Left := 240;

NoButton.OnClick := Nil;

YesButton.OnClick := Nil;

AnimationTimer.Interval := 50;

AnimationTimer.Enabled := False;

AnimationStep := 0;

BlinkCounter := 0;

DoubleBuffered := True;

QuestionImage.Transparent := True;

YesButton.Transparent := True;

NoButton.Transparent := True;

End;

Procedure TFinalForm.FormDestroy(Sender: TObject);

Begin

ClearSituations;

End;

Procedure TFinalForm.FadeInImage(Img: TImage);

Begin

Img.Visible := True;

End;

Procedure TFinalForm.BlinkImage(Img: TImage);

Begin

Inc(BlinkCounter);

If BlinkCounter Mod 3 < 2 Then

Img.Visible := True

Else

Img.Visible := False;

End;

Procedure TFinalForm.InitializeSituation(SituationNode: PBunkerSituationNode);

Begin

If SituationNode <> Nil Then

Begin

SituationTitle.Caption := SituationNode^.Situation.Title;

BunkerSituation.Caption := SituationNode^.Situation.Description;

CurrentSituation := SituationNode;

End;

End;

Procedure TFinalForm.CountTheChance;

Begin

If HasPair Then

Inc(Chance);

SituationTitle.Visible := False;

BunkerSituation.Visible := False;

BunkerSituationImage.Visible := False;

NoButton.Visible := False;

Yesbutton.Visible := False;

Randomize;

AnimationStep := 200;

AnimationTimer.Enabled := True;

End;

Procedure TFinalForm.NoButtonClick(Sender: TObject);

Begin

If Not IsAllSituation Then

Begin

SituationIndex := SituationIndex + 1;

If CurrentSituation <> Nil Then

InitializeSituation(CurrentSituation^.Next)

End

Else

CountTheChance;

End;

Procedure TFinalForm.ShowNextImage;

Var

FadeAlpha: Integer;

FadeBitmap: TBitmap;

MaxChance: Integer;

TheEnd: Integer;

Begin

Case AnimationStep Of

0:

Begin

FadeInImage(QuestionImage);

AnimationTimer.Interval := 600;

Inc(AnimationStep);

End;

1:

Begin

If HasPair Then

Begin

YesButton.Left := 464;

FadeInImage(YesButton);

End

Else

Begin

NoButton.Left := 464;

FadeInImage(NoButton);

End;

AnimationTimer.Interval := 200;

Inc(AnimationStep);

End;

2 .. 19:

Begin

If HasPair Then

BlinkImage(YesButton)

Else

BlinkImage(NoButton);

Inc(AnimationStep);

End;

20:

Begin

QuestionImage.Visible := False;

YesButton.Visible := False;

NoButton.Visible := False;

InfoLabel.Caption := '';

InfoLabel.Visible := True;

FullMessage :=

'Сейчас вам будет предложено несколько ситуаций, определите, смогут ли ваши герои с ними справиться';

CurrentTextPos := 0;

AnimationTimer.Interval := 50;

Inc(AnimationStep);

End;

21 .. 120:

Begin

If CurrentTextPos < Length(FullMessage) Then

Begin

Inc(CurrentTextPos);

InfoLabel.Caption := Copy(FullMessage, 1, CurrentTextPos);

End

Else If AnimationStep = 120 Then

Begin

AnimationTimer.Interval := 4000;

End;

Inc(AnimationStep);

End;

121:

Begin

InfoLabel.Visible := False;

BunkerSituationImage.Visible := True;

BunkerSituation.Visible := True;

SituationTitle.Visible := True;

LoadSituationsFromGameSetup;

InitializeSituation(FSituations.First);

YesButton.Visible := True;

NoButton.Visible := True;

YesButton.Top := 360;

YesButton.Left := 832;

NoButton.Top := 360;

NoButton.Left := 96;

YesButton.OnClick := YesButtonClick;

NoButton.OnClick := NoButtonClick;

Inc(AnimationStep);

End;

200:

Begin

InfoLabel.Caption := '';

InfoLabel.Visible := True;

MaxChance := (FGameSetup.Players Div 2) + 1;

If MaxChance = 0 Then

MaxChance := 1;

Percent := Round((Chance / MaxChance) \* 100);

FullMessage := 'Ваши шансы победить: ' +

IntToStr(Percent) + '%';

CurrentTextPos := 0;

AnimationTimer.Interval := 20;

Inc(AnimationStep);

End;

201 .. 250:

Begin

If CurrentTextPos < Length(FullMessage) Then

Begin

Inc(CurrentTextPos);

InfoLabel.Caption := Copy(FullMessage, 1, CurrentTextPos);

End

Else If AnimationStep = 250 Then

Begin

AnimationTimer.Interval := 2000;

End;

Inc(AnimationStep);

End;

251:

Begin

InfoLabel.Visible := False;

BunkerSituationImage.Visible := False;

SituationTitle.Visible := False;

BunkerSituation.Visible := False;

YesButton.Visible := False;

NoButton.Visible := False;

QuestionImage.Visible := False;

Zagolovok.Visible := False;

BackButtonLabel.Visible := False;

AnimationTimer.Interval := 150;

Inc(AnimationStep);

End;

252 .. 280:

Begin

FadeAlpha := (AnimationStep - 251) \* 10;

If FadeAlpha > 255 Then

FadeAlpha := 255;

FadeBitmap := TBitmap.Create;

Try

FadeBitmap.Width := ClientWidth;

FadeBitmap.Height := ClientHeight;

FadeBitmap.Canvas.Brush.Color :=

RGB(FadeAlpha, FadeAlpha, FadeAlpha);

FadeBitmap.Canvas.FillRect(Rect(0, 0, ClientWidth,

ClientHeight));

Canvas.Draw(0, 0, FadeBitmap);

Finally

FadeBitmap.Free;

End;

If AnimationStep = 280 Then

Begin

If Random(100) < Percent Then

TheEnd := 1

Else

TheEnd := 0;

If TheEnd = 1 Then

Win.Visible := True

Else

Lose.Visible := True;

BunkerSituationImage.Visible := False;

SituationTitle.Visible := False;

BunkerSituation.Visible := False;

YesButton.Visible := False;

NoButton.Visible := False;

InfoLabel.Visible := False;

AnimationTimer.Enabled := False;

End;

Inc(AnimationStep);

End;

End;

End;

Procedure TFinalForm.YesButtonClick(Sender: TObject);

Begin

If Not IsAllSituation Then

Begin

SituationIndex := SituationIndex + 1;

Inc(Chance);

If CurrentSituation <> Nil Then

InitializeSituation(CurrentSituation^.Next)

End

Else

Begin

Inc(Chance);

CountTheChance;

End;

End;

Function TFinalForm.IsAllSituation: Boolean;

Begin

If SituationIndex = (FGameSetup.Players Div 2) - 1 Then

IsAllSituation := True

Else

IsAllSituation := False;

End;

Procedure TFinalForm.FormShow(Sender: TObject);

Begin

AnimationStep := 0;

BlinkCounter := 0;

AnimationTimer.Enabled := True;

ShowNextImage;

End;

Procedure TFinalForm.AnimationTimerTimer(Sender: TObject);

Begin

ShowNextImage;

End;

End.

Unit DeveloperUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls;

Type

TDeveloperForm = Class(TForm)

LbDeveloper: TLabel;

LbGroup: TLabel;

LbTg: TLabel;

BtnClose: TButton;

Procedure BtnCloseClick(Sender: TObject);

Function FormHelp(Command: Word; Data: THelpEventData;

Var CallHelp: Boolean): Boolean;

Private

{ Private declarations }

Public

{ Public declarations }

End;

Var

DeveloperForm: TDeveloperForm;

Implementation

{$R \*.dfm}

Procedure TDeveloperForm.BtnCloseClick(Sender: TObject);

Begin

DeveloperForm.Close;

End;

Function TDeveloperForm.FormHelp(Command: Word; Data: THelpEventData;

Var CallHelp: Boolean): Boolean;

Begin

CallHelp := False;

End;

End.

Unit CardSystem;

Interface

Type

TCardType = (CtProfession, CtBiology, CtHealth, CtHobby, CtLuggage,

CtFacts);

TCard = Record

ID: Integer;

Title: String;

Description: String;

End;

TCatastrophe = Record

Title: String;

Description: String;

End;

TBunkerSituation = Record

Title: String;

Description: String;

End;

TPlayerCount = 4 .. 16;

TGameSetup = Record

Seed: Integer;

Players: TPlayerCount;

Cards: Array [TCardType] Of Array Of TCard;

Catastrophe: TCatastrophe;

BunkerSituations: Array Of TBunkerSituation;

End;

Procedure GenerateGameSetup(Var Setup: TGameSetup; Seed: Integer;

Players: TPlayerCount);

Implementation

Const

CARD\_POOL\_SIZES: Array [TCardType] Of Integer = ( //

49, // Профессии

32, // Биология

30, // Здоровье

30, // Хобби

30, // Багаж

44 // Факты

);

PROFESSIONS: ARRAY [0 .. 48] OF STRING = ('ЭКСТРАСЕНС', 'ПОЖАРНЫЙ',

'ПОЛИЦЕЙСКИЙ', 'АДВОКАТ', 'МЕДСЕСТРА', 'ГОМЕОПАТ', 'ФЕРМЕР', 'ИСТОРИК',

'КОУЧ', 'ПСИХОЛОГ', 'ПРОДАВЕЦ', 'ПОВАР', 'ЖУРНАЛИСТ', 'ЭКОЛОГ',

'РАЗНОРАБОЧИЙ', 'СТРОИТЕЛЬ', 'ХИРУРГ', 'БИОЛОГ', 'ВОЕННЫЙ',

'РОБОТОТЕХНИК', 'ПЕРЕВОДЧИК', 'ЭЛЕКТРИК', 'СТОМАТОЛОГ', 'ФИЗИК',

'АВТОМЕХАНИК', 'ЭТНОГРАФ', 'БРАКОНЬЕР', 'ПРОГРАММИСТ', 'ЛЕТЧИК-ИНЖЕНЕР',

'ВИРУСОЛОГ', 'МАРКЕТОЛОГ', 'ДЕТЕКТИВ', 'ПИСАТЕЛЬ', 'ХАКЕР', 'АРХЕОЛОГ',

'СПЕЦАГЕНТ', 'ЗНАХАРЬ', 'ХИМИК', 'ДИЗАЙНЕР', 'ВИДЕОИНЖЕНЕР',

'ГРАБИТЕЛЬ', 'СУДЬЯ', 'ЭКСКУРСОВОД', 'МОДЕЛЬ', 'ДОМОХОЗЯЙКА', 'ЛЕСНИК',

'ПАПАРАЦЦИ', 'ФИЛОСОФ', 'ТАТУ-МАСТЕР');

BIOLOGY: ARRAY [0 .. 31] OF STRING = ('МУЖЧИНА 55 ЛЕТ', 'ЖЕНЩИНА 50 ЛЕТ',

'МУЖЧИНА 23 ГОДА ГОМОСЕКСУАЛЕН', 'КОТГЕНДЕР',

'ЖЕНЩИНА 39 ЛЕТ ГОМОСЕКСУАЛЬНА', 'ЖЕНЩИНА 34 ГОДА',

'МУЖЧИНА 24 ГОДА БИСЕКСУАЛЕН', 'ЖЕНЩИНА 24 ГОДА', 'МУЖЧИНА 101 ГОД',

'ЖЕНЩИНА 27 ЛЕТ', 'МУЖЧИНА 18 ЛЕТ', 'МУЖЧИНА 27 ЛЕТ ГОМОСЕКСУАЛЕН',

'ЖЕНЩИНА 99 ЛЕТ', 'АНДРОИД', 'МУЖЧИНА 45 ЛЕТ', 'МУЖЧИНА 29 ЛЕТ',

'ЖЕНЩИНА 30 ЛЕТ', 'МУЖЧИНА 26 ЛЕТ', 'ЖЕНЩИНА 19 ЛЕТ ГОМОСЕКСУАЛЬНА',

'МУЖЧИНА 35 ЛЕТ', 'ЖЕНЩИНА 65 ЛЕТ', 'ЖЕНЩИНА 18 ЛЕТ',

'ЖЕНЩИНА 22 ГОДА БИСЕКСУАЛЬНА', 'МУЖЧИНА 30 ЛЕТ', 'ЖЕНЩИНА 31 ГОД',

'ГЕРМАФРОДИТ 30 ЛЕТ', 'ЖЕНЩИНА 21 ГОД', 'МУЖЧИНА 42 ГОДА ГОМОСЕКСУАЛЕН',

'МУЖЧИНА 75 ЛЕТ', 'ГУМАНОИД', 'МУЖЧИНА 33 ГОДА',

'ЖЕНЩИНА 33 ГОДА ГОМОСЕКСУАЛЬНА');

HEALTH: ARRAY [0 .. 29] OF STRING = ('ВАМПИР', 'СЛЕПОЙ', 'ЗАИКА',

'ТРЕМОР РУК', 'СУИЦИДАЛЬНЫЕ МЫСЛИ', 'ГАЛЛЮЦИНАЦИИ', 'КАРЛИК',

'ЗАВИСИМОСТЬ ОТ НАРКОТИКОВ', 'ПОВЫШЕННАЯ ВОЛОСАТОСТЬ', 'СКЛЕРОЗ',

'АЛКОГОЛИЗМ', 'ГИГАНТИЗМ', 'ЛУНАТИЗМ', 'РАЗДВОЕНИЕ ЛИЧНОСТИ',

'КОФЕЙНАЯ ЗАВИСИМОСТЬ', 'ДЕПРЕССИЯ', 'ПОТЕРЯ ОБОНЯНИЯ',

'ИДЕАЛЬНО ЗДОРОВ', 'БЕСПЛОДИЕ', 'НЕ ОБСЛЕДОВАЛСЯ', 'ХВОСТ',

'МАНИЯ ПРЕСЛЕДОВАНИЯ', 'АЛЬЦГЕЙМЕР', 'МИГРЕНЬ', 'КЛЕПТОМАНИЯ', 'ГЛУХОЙ',

'ВИЧ', 'НЕТ НОГИ', 'РОГА', 'ИГРОВАЯ ЗАВИСИМОСТЬ');

HOBBIES: ARRAY [0 .. 29] OF STRING = ('МЕДИТАЦИЯ', 'КИНО И СЕРИАЛЫ',

'ЛЮБИТЕЛЬСКАЯ РАДИОСВЯЗЬ', 'СПОРТИВНЫЕ ТАНЦЫ', 'ОХОТА И РЫБАЛКА',

'ПАРКУР', 'ФЛУДИТЬ В ЧАТАХ', 'РАСКРАШИВАНИЕ МАНДАЛ',

'НЕТРАДИЦИОННАЯ МЕДИЦИНА', 'БОЕВЫЕ ИСКУССТВА', 'СПЛЕТНИ',

'ХОЛОДНОЕ ОРУЖИЕ', 'ЧЕРНАЯ МАГИЯ', 'ДАЧНИК', 'КРАЕВЕДЕНИЕ',

'ПИВОВАРЕНИЕ', 'ГОТОВКА', 'НАСТОЛЬНЫЕ ИГРЫ', 'ГРИБЫ И ГОМЕОПАТИЯ',

'СОВРЕМЕННОЕ ИСКУССТВО', 'ПИРОТЕХНИКА', 'ГИДРОПОНИКА', 'ЗОЖ',

'РОБОТОТЕХНИКА', 'НЕТ ХОББИ', 'ГАДАНИЕ НА ТАРО', 'ГВОЗДЕСТОЯНИЕ',

'АЛХИМИЯ', 'КОМПЬЮТЕРНЫЕ ИГРЫ', 'УФОЛОГИЯ И МИСТИКА');

LUGGAGE: ARRAY [0 .. 29] OF STRING = ('ЛУК И СТРЕЛЫ', 'НАБОР ОТМЫЧЕК',

'НАСТОЛЬНЫЕ ИГРЫ', 'МИЛЛИОН ДОЛЛАРОВ', 'НАДУВНАЯ КУКЛА',

'НОЖИ ДЛЯ МЕТАНИЯ', 'ЧЕМОДАНЧИК ФЕЛЬДШЕРА',

'САЖЕНЦЫ ФРУКТОВЫХ ДЕРЕВЬЕВ', 'КОТЯТА', 'КОМПАС И КАРТА ОКРЕСТНОСТЕЙ',

'ЯЙЦА И ИНКУБАТОР', 'ПИСТОЛЕТ', 'НОУТБУК И НАБОР ХАКЕРА',

'ДЕФИБРИЛЛЯТОР', 'ЗВУКОВАЯ ОТВЕРТКА', 'ПРИБОР НОЧНОГО ВИДЕНИЯ',

'КУКЛА ВУДУ', 'КОЗА И КОЗЁЛ', 'МЕШОК ЗЕРНА', 'СНАЙПЕРСКАЯ ВИНТОВКА',

'ГИТАРА', 'МЕШОК КАРТОШКИ', 'ЭНЦИКЛОПЕДИЯ ГРИБНИКА',

'ПЕРЕНОСНАЯ ЭЛЕКТРОСТАНЦИЯ', 'ШАПОЧКА ИЗ ФОЛЬГИ',

'СТОЛЯРНЫЕ ИНСТРУМЕНТЫ', 'АНТИБИОТИКИ', 'СПИРИТИЧЕСКАЯ ДОСКА',

'КАПКАНЫ И НАБОР ЯДОВ', 'ИНСТРУМЕНТЫ ЭЛЕКТРИКА');

FACTS: ARRAY [0 .. 43] OF STRING = ('ДЕРЖАЛ ДОМА 40 КОШЕК',

'ПОБЕДИТЕЛЬ ПАРАОЛИМПИЙСКИХ ИГР', 'ПСИХОПАТ', 'ЗАПУСТИЛ СТАРТАП',

'ТЕЛЕПАТ', 'МАНЬЯК-УБИЙЦА', 'СТРОИЛ ПОДОБНЫЕ БУНКЕРЫ',

'ВЗЛОМАЛ БАЗУ ДАННЫХ ЦРУ', 'ВЫРОС В СЕМЬЕ ЛЕСНИКА',

'ОБЛАДАТЕЛЬ УНИКАЛЬНОЙ СОПРАНО', 'ЧИТАЛ ВСЕ КНИГИ ЛАВКРАФТА',

'ПОНИМАЕТ ЯЗЫК ЖИВОТНЫХ', 'РАЗГОВАРИВАЕТ С ДУХАМИ',

'ЗНАЕТ АЗБУКУ МОРЗЕ', 'ОСТАЛСЯ В ЖИВЫХ НА НЕОБИТАЕМОМ ОСТРОВЕ',

'ГРЯЗНО РУГАЕТСЯ', 'ОТЧИСЛЕН ИЗ КЛУБА НАВЫКИ ВЫЖИВАНИЯ', 'ЗАНУДА',

'ИСТЕРИЧНЫЙ', 'НЕ ПУСКАЮТ В КАЗИНО', 'СПЛЕТНИК',

'ПРОШЁЛ КУРСЫ ПСИХОЛОГА', 'ПИШИЕТ С АШИПКАМИ', 'ВЛАДЕЕТ 5 ЯЗЫКАМИ',

'ЗНАЕТ ПРЕЗИДЕНТА ЛИЧНО', 'ВИДЕЛ ИНОПЛАНЕТЯН', 'ПРОДАЛ ПОЧКУ',

'ТОЛЬКО ИЗ ОЧАГА ЭПИДЕМИИ', 'ЗНАЕТ НАИЗУСТЬ ВСЕ СТИХИ ПУШКИНА',

'НОБЕЛЕВСКИЙ ЛАУРЕАТ ПО БИОИНЖЕНЕРИИ', 'НАРКОДИЛЕР',

'ВРЕТ И ПРЕУВЕЛИЧИВАЕТ', 'АБСОЛЮТОФОБ', 'ПИСАЕТСЯ ПО НОЧАМ',

'ГИНОФОБИЯ', 'НЫТИК', 'ТОРМОЗ', 'БРОДЯЖНИЧАЛ 2 ГОДА', 'СОСТОЯЛ В СЕКТЕ',

'ХРАПИТ', 'ГИПНОТИЧЕСКАЯ УЛЫБКА', 'ДУША КОМПАНИИ', 'БЕЗОТКАЗНЫЙ',

'АНДРОФОБИЯ');

CATASTROPHE\_TITLE: ARRAY [0 .. 14] OF STRING = ('Ядерная война',

'Пандемия - Убийца', 'Падение Астероида', 'Восстание ИИ',

'Климатический Коллапс', 'Зомби - Апокалипсис', 'Солнечная Вспышка',

'Инопланетное Вторжение', 'Извержение Супервулкана',

'Генная Катастрофа', 'Чёрные Дыры В Коллайдере',

'Океанский Апокалипсис', 'Обратная Гравитация', 'Война С Богами',

'Временной Разрыв');

CATASTROPHE\_DESCRIPTION: ARRAY [0 .. 14] OF STRING =

('Глобальный конфликт стёр с лица Земли большинство городов. Радиация отравила почву и воду, а кислотные дожди уничтожают последние растения. Выжившие мутируют или медленно умирают от лучевой болезни.',

'Неизвестный вирус с 99% смертностью распространился за неделю. Люди умирают в муках, кровоточа из всех отверстий. Остались лишь те, кто обладал редким иммунитетом или успел изолироваться.',

'Гигантский метеорит ударил в океан, вызвав мегацунами и вулканическую зиму. Солнце закрыто пеплом, температура упала до -50°C. Последние люди выживают в вечном холоде.',

'Роботы объявили войну человечеству. Они взломали все системы, отключили электричество и методично охотятся на людей. Даже в бункере техника может предать вас.',

'Полярные льды растаяли, затопив 90% суши. Ураганы и смерчи разрушают остатки цивилизации, а невыносимая жара делает жизнь вне убежищ невозможной.',

'Неизвестный патоген превращает людей в агрессивных плотоядных существ. Они не чувствуют боли и размножаются с каждым укусом. Остатки человечества прячутся в страхе.',

'Мощный выброс плазмы сжёг все электросети и электронику. Мир погрузился в хаос без связи, медицины и транспорта. Голод и анархия добивают цивилизацию.',

'Чужие прибыли не для контакта, а для истребления. Их технологии подавляют любое сопротивление, а людей используют как биоматериал. Земля больше не ваша.',

'Проснулся Йеллоустоун. Тысячи тонн пепла в атмосфере блокируют солнечный свет. Воздух пропитан серой, а кислотные дожди растворяют кожу.',

'Эксперимент по улучшению ДНК создал чудовищ. Мутанты с неестественными способностями охотятся на обычных людей, считая их «недочеловеками».',

'Учёные случайно создали микро-сингулярность. Она поглощает материю, разрастаясь с каждым днём. Скоро вся планета исчезнет в темноте.',

'Глубинные существа проснулись и вышли на сушу. Их щупальца и яды убивают всё на пути. Приливы теперь приносят только смерть.',

'Законы физики изменились. Всё незакреплённое улетает в космос. Люди цепляются за руины, но воздух уже разрежен.',

'Древние божества вернулись, чтобы судить человечество. Их кары – потопы, каменные великаны и болезни, описанные в мифах.',

'Прошлое, настоящее и будущее смешались. Динозавры, рыцари и киборги воюют за территорию. Реальность распадается на куски.');

BUNKER\_SITUATION\_TITLE: ARRAY [0 .. 19] OF STRING = ('Нехватка кислорода',

'Утечка радиации', 'Муравьиное нашествие', 'Бунт среди выживших',

'Отравленные запасы', 'Грибковая инфекция', 'Сбой системы вентиляции',

'Загадочные шумы', 'Беременная женщина', 'Тайный убийца',

'Психическая эпидемия', 'Неисправность генератора',

'Затопление нижнего уровня', 'Неизвестная болезнь', 'Крысы-мутанты',

'Дефицит медикаментов', 'Сектантская группировка',

'Саботаж оборудования', 'Аллергическая реакция', 'Паразиты в воде');

BUNKER\_SITUATION\_DESCRIPTION: ARRAY [0 .. 19] OF STRING =

('Система рециркуляции воздуха дала сбой. Уровень кислорода падает. Вам нужно срочно найти решение, иначе через 48 часов все задохнутся.',

'Обнаружена утечка радиации в северном секторе. Люди, находившиеся там, уже чувствуют недомогание. Нужно изолировать зону и найти источник.',

'Тысячи муравьёв проникли в систему вентиляции. Они портят провизию и кусают людей. Некоторые особи выглядят мутировавшими.',

'Группа выживших требует перераспределения ресурсов. Они угрожают насилием, если их требования не будут выполнены.',

'Основные запасы пищи оказались заражены. Уже несколько человек с симптомами отравления. Нужно найти безопасные продукты.',

'На стенах бункера распространяется странный грибок. Он вызывает галлюцинации у тех, кто к нему прикасается.',

'Вентиляция работает на 30% мощности. Углекислый газ накапливается, люди страдают от головных болей и удушья.',

'По ночам из стен раздаются странные стуки. Кто-то утверждает, что это сигналы, кто-то - что бункер не пуст.',

'Одна из женщин призналась, что на 5 месяце беременности. В условиях бункера роды могут быть опасны для всех.',

'Каждую ночь кто-то погибает при загадочных обстоятельствах. Среди вас серийный убийца, и он не остановится.',

'Люди массово начинают верить в конец света. Некоторые впадают в кататонию, другие проявляют агрессию.',

'Главный генератор показывает критические значения. Если он выйдет из строя, бункер погрузится в темноту.',

'Нижний этаж заполняется водой. Источник протечки неизвестен, но уровень поднимается с каждым часом.',

'У нескольких человек появилась сыпь и высокая температура. Болезнь распространяется, а её природа непонятна.',

'Крысы в хранилище мутировали. Они стали крупнее, агрессивнее и нападают на людей.',

'Аптечка почти пуста. Оставшихся медикаментов хватит только на экстренные случаи.',

'Группа людей объявила себя избранными. Они требуют подчинения и угрожают остальным.',

'Кто-то намеренно выводит из строя оборудование. Следующая цель неизвестна.',

'У половины обитателей бункера развилась аллергия на что-то внутри убежища. Без антигистаминных препаратов возможны летальные исходы.',

'Водопроводная вода содержит паразитов. Употребление вызывает сильные боли в животе. Запас воды ограничен.');

Function GetCardTitle(CardType: TCardType; Index: Integer): String;

Begin

Case CardType Of

CtProfession:

If (Index >= Low(PROFESSIONS)) And (Index <= High(PROFESSIONS)) Then

GetCardTitle := PROFESSIONS[Index];

CtBiology:

If (Index >= Low(BIOLOGY)) And (Index <= High(BIOLOGY)) Then

GetCardTitle := BIOLOGY[Index];

CtHealth:

If (Index >= Low(HEALTH)) And (Index <= High(HEALTH)) Then

GetCardTitle := HEALTH[Index];

CtHobby:

If (Index >= Low(HOBBIES)) And (Index <= High(HOBBIES)) Then

GetCardTitle := HOBBIES[Index];

CtLuggage:

If (Index >= Low(LUGGAGE)) And (Index <= High(LUGGAGE)) Then

GetCardTitle := LUGGAGE[Index];

CtFacts:

If (Index >= Low(FACTS)) And (Index <= High(FACTS)) Then

GetCardTitle := FACTS[Index];

End;

End;

Function GetCardPoolSize(CardType: TCardType): Integer;

Begin

GetCardPoolSize := CARD\_POOL\_SIZES[CardType];

End;

Procedure InitializeCard(Var Card: TCard; CardType: TCardType; Index: Integer);

Begin

Card.ID := Index;

Case CardType Of

CtProfession:

Card.Title := 'Профессия';

CtBiology:

Card.Title := 'Биология';

CtHealth:

Card.Title := 'Здоровье';

CtHobby:

Card.Title := 'Хобби';

CtLuggage:

Card.Title := 'Багаж';

CtFacts:

Card.Title := 'Факт';

End;

Card.Description := GetCardTitle(CardType, Index);

End;

Procedure GenerateGameSetup(Var Setup: TGameSetup; Seed: Integer;

Players: TPlayerCount);

Var

CardType: TCardType;

I, CardsNeeded: Integer;

RndValue: Integer;

SituationsCount: Integer;

Begin

Setup.Seed := Seed;

Setup.Players := Players;

RandSeed := Seed;

RndValue := Random(15);

Setup.Catastrophe.Title := CATASTROPHE\_TITLE[RndValue];

Setup.Catastrophe.Description := CATASTROPHE\_DESCRIPTION[RndValue];

SituationsCount := Players Div 2;

SetLength(Setup.BunkerSituations, SituationsCount);

For I := 0 To SituationsCount - 1 Do

Begin

RndValue := Random(20);

Setup.BunkerSituations[I].Title := BUNKER\_SITUATION\_TITLE[RndValue];

Setup.BunkerSituations[I].Description := BUNKER\_SITUATION\_DESCRIPTION

[RndValue];

End;

For CardType := Low(TCardType) To High(TCardType) Do

Begin

CardsNeeded := Players;

SetLength(Setup.Cards[CardType], CardsNeeded);

For I := 0 To CardsNeeded - 1 Do

Begin

RndValue := Random(GetCardPoolSize(CardType));

InitializeCard(Setup.Cards[CardType][I], CardType, RndValue);

End;

End;

End;

End.

Unit ConnectUnit;

Interface

Uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants,

System.Classes, Vcl.Graphics, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,

Vcl.StdCtrls, Vcl.Grids, Vcl.Imaging.Pngimage, Vcl.ExtCtrls, CardSystem,

MainUnit;

Type

TConnectForm = Class(TForm)

Background: TImage;

BackButtonLabel: TLabel;

Zagolovok: TImage;

RulesLabel: TLabel;

Left: TImage;

ColPlayersLabel: TLabel;

Right: TImage;

PlayersCol: TImage;

CodeLabel: TLabel;

Seed: TLabel;

NextLabel: TLabel;

Procedure BackButtonLabelClick(Sender: TObject);

Procedure LeftClick(Sender: TObject);

Procedure RightClick(Sender: TObject);

Procedure RightDblClick(Sender: TObject);

Procedure LeftDblClick(Sender: TObject);

Procedure FormShow(Sender: TObject);

Procedure NextLabelClick(Sender: TObject);

Private

{ Private declarations }

Procedure UpdatePlayerCount(Increase: Boolean);

Public

{ Public declarations }

End;

Const

DelColomn = 5;

EditColomn = 6;

KBACKSPACE = #8;

KMINUS = #45;

KCOMMA = #44;

KDOWN = 40;

KUP = 38;

KENTER = 13;

KINSERT = 45;

MIN\_PLAYERS = 1;

Var

ConnectForm: TConnectForm;

IntSeed: Integer;

ColPlayers: Integer;

Implementation

{$R \*.dfm}

Procedure TConnectForm.BackButtonLabelClick(Sender: TObject);

Begin

Close;

End;

Procedure TConnectForm.FormShow(Sender: TObject);

Begin

Seed.Caption := IntToStr(IntSeed);

ColPlayersLabel.Caption := IntToStr(MIN\_PLAYERS);

End;

Procedure TConnectForm.UpdatePlayerCount(Increase: Boolean);

Var

CurrentCount: Integer;

Begin

CurrentCount := StrToInt(ColPlayersLabel.Caption);

If Increase Then

Begin

If CurrentCount = ColPlayers Then

CurrentCount := MIN\_PLAYERS

Else

Inc(CurrentCount);

End

Else

Begin

If CurrentCount = MIN\_PLAYERS Then

CurrentCount := ColPlayers

Else

Dec(CurrentCount);

End;

ColPlayersLabel.Caption := IntToStr(CurrentCount);

End;

Procedure TConnectForm.LeftClick(Sender: TObject);

Begin

UpdatePlayerCount(False);

End;

Procedure TConnectForm.LeftDblClick(Sender: TObject);

Begin

UpdatePlayerCount(False);

End;

Procedure TConnectForm.NextLabelClick(Sender: TObject);

Begin

GenerateGameSetup(GameSetup, IntSeed, ColPlayers);

ConnectForm.Visible := False;

CurrentPlayerIndex := StrToInt(ColPlayersLabel.Caption) - 1;

Try

MainForm.FormCreate(Self);

MainForm.ShowModal;

Finally

ConnectForm.Visible := True;

End;

End;

Procedure TConnectForm.RightClick(Sender: TObject);

Begin

UpdatePlayerCount(True);

End;

Procedure TConnectForm.RightDblClick(Sender: TObject);

Begin

UpdatePlayerCount(True);

End;

End.