Оглавление

Цель работы…………………………………………………………...….….........2

Задачи……………………………………………………………………………...2

1.Техническое задание……………………………………………………………3

Введение……………………………………………………………………….…. 4

1.1Наименование продукта…………………………………………..……….....4

1.2 Краткая характеристика области применения объекта................................4

2. Основание для разработки…………………………………………………......4

2.1 Документ, на основании которого ведется разработка …………………...4

2.2 Организация, утвердившая документ……………………………………....4

[3. Назначение разработки 4](#_Toc503503631)

[3.1 Функциональное назначение 4](#_Toc503503632)

[3.2 Эксплуатационное назначение 4](#_Toc503503633)

4. Требования к разработке…………………………………………………….....5

[4.1 Требования к функциональным характеристикам 5](#_Toc503503634)

4.2 Требования к надежности…………………………………………………...5

4.3 Требования к составу и параметрам технических средств………………..6

4.4 Требования к информационной и программной совместимости…………6

5. Требования к программной документации……………………………….….. 6

[6. Стадии и этапы разработки 6](#_Toc503503636)

[6.1 Стадии разработки 6](#_Toc503503637)

[6.2 Этапы разработки 7](#_Toc503503638)

[6.3 Содержание работ по этапам 7](#_Toc503503639)

[Описание структуры программы 8](#_Toc503503640)

Руководство пользователя…………………………………………………........16

Приложение………………………………………………………………………21

# Цель работы

Целью работы является получение и отработка теоретических знаний и практических навыков по разработке приложений на языке высокого уровня C#.

Задачи

* Написание технического задания будущего приложения;
* Проектирование архитектуры приложения;
* Программная реализация алгоритмов приложения;
* Реализация интерфейса приложения;
* Составление отчета по результатам учебной практики.

**Министерство образования и науки Российской Федерации**

Федеральное государственное автономное образовательное учреждение

высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт Кибернетики

Направление Информационные системы и технологии

|  |  |
| --- | --- |
|  |  |
| Компьютерная игра «Тамагочи»  наименование вида ИС  Техническое задание | |
| **СОГЛАСОВАНО**  Должность  Старший преподаватель кафедры ВТ Лепустин А.В.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. | **РАЗРАБОТЧИК**  Студент гр.8И52  Смирнов П.О.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г. |
| Томск 2018 | |

Введение

1.1 Наименование продукта

Однопользовательская компьютерная игра в жанре аркадных игр, Tamagotchi.

**1.2 Краткая характеристика области применения**

Игра «Tamagotchi» представляет собой однопользовательскую игру. Предназначена для развлечения пользователей.

**2. Основание для разработки**

**2.1 Документ, на основании которого ведется разработка**

Задание на учебную практику.

**2.1 Организация, утвердившая документ**

Томский политехнический университет

# 3. Назначение разработки

## **3.1 Функциональное назначение**

Программа относится к разряду развлекательных. Она дает любителям компьютерных игр провести свободное время.

## **3.2 Эксплуатационное назначение**

Программа может эксплуатироваться в домашних условиях. Конечные потребители – рядовые пользователи, не нуждающиеся в дополнительном инструктаже и обучении.

**4. Требования к разработке**

## **4.1 Требования к функциональным характеристикам**

### **4.1.1** **Требования к составу выполняемых функций**

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* Создание новой игры;
* Программа должна осуществлять управление с помощью клавиатуры/мыши;
* Программа должна иметь игровое меню;
* Запись результатов в таблицу рейтинга, где будет учитываться количество набранных очков;
* Отображение названия программы;

Игра должна представлять из себя приложение, состоящее из следующих окон:

1. Играть
2. Лучший результат
3. Выход

Навигация по меню должна осуществляться с помощью мыши. Для выбора пункта меню пользователю необходимо навести курсор на нужный пункт и нажать левую кнопку мыши.

#### **4.1.1.1 Главное меню**

В главном меню игроку должны предлагаться такие функции, как начать игру, просмотреть лучшие результаты и выйти из игры.

**4.2 Требования к надежности**

Программа не должна самостоятельно завершаться или выдавать ошибку при корректных действиях пользователя. Отказы программы возможны вследствие некорректных действий оператора(пользователя) при взаимодействии с операционной системой. Во избежание возникновение отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

**4.3 Требования к составу и параметрам технических средств**

* Процессор Intel или совместимый;
* Объем свободной оперативной памяти 16 Мб;
* Объем необходимой памяти на жестком диске 50 Мб;
* Монитор с поддерживаемым разрешением 1024 x 768;
* Клавиатура / Мышь;

**4.4 Требование к информационной и программной совместимости**

Для функционирования игры «Тамагочи» необходимо наличие операционной системы Microsoft Windows 7 и новее, Microosoft .NET Framework 2. Язык интерфейса - русский.

**5. Требования к программной документации**

Предварительный состав программной документации:

* “Техническое задание”, текст программы с комментариями, описание алгоритма и программы.
* “Руководство пользователя”.

# 

6. Стадии и этапы разработки

## **6.1 Стадии разработки**

Разработка должна быть проведена в три стадии:

1. разработка технического задания;
2. рабочее проектирование;
3. внедрение.

## **6.2 Этапы разработки**

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1. разработка программы;
2. разработка программной документации;
3. испытания программы.

На стадии внедрения должен быть выполнен этап разработки – подготовка и передача программы.

## **6.3 Содержание работ по этапам**

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. постановка задачи;
2. определение и уточнение требований к техническим средствам;
3. определение требований к программе;
4. определение стадий, этапов и сроков разработки программы;
5. выбор языков программирования;
6. согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию и отладке программы.

# Описание структуры программы

Описание логики переходов между уровнями, завершения игры, запуска меню, новой игры и взаимодействия между ними осуществлял основной класс игры – статический класс GameManager.

Логика игры осуществляется абстрактным классом GUI. Он осуществляет подписку и отписку функций класса GameManager на методы, отображение игрового поля, а также контролирует процесс наведения и нажатия указателя мыши на элементы игрового пространства.

Основной игровой процесс происходит на игровом поле размерностью 560 на 330 pixels. При запуске программы открывается окно Menu на котором распологаются три кнопки Play, Records и Exit, элементы класса SelectButton. При нажатии пользователя на кнопку происходит начало игры. При начале игры запускается пользовательский таймер, объект класса OurTimer, и создается объект класса Pet, характеристики которого пользователь меняет по ходу игры, нажимая на кнопки Play, Eat, Sleep, также объекты класса SelectButton. Также на игровом окне расположены текстовые поля Satiety(сытость), Energy, Mood(настроение), Coins (игровые монеты) и Time(время данной сессии), объекты класса LabelText .В ходе игры с помощью метода Tick класса OurTimer, происходит постепенное уменьшение характеристик питомца. Пользователь при нажатии на кнопки Play, Eat, Sleep увеличивает характеристики питомца Mood(настроение), Satiety(сытость), Energy соответственно. При нажатии на кнопку Eat пользователь тратит Coins(монеты), валидацию данных проводит метод Buy, класса Shop. Пользователь увеличивает количество монет при нажатии кнопки Play. Логика игры устроена таким образом, что кол-во потраченных монет будет всегда больше кол-ва полученных, тем самым подводя игру к концу. Каждые 20 секунд увеличивается коэффициент уменьшения характеристик питомца, который делает игру сложнее.

При завершении игровой сессии вызывается метод EndSession класса GameMenu, который вызывают создание уведомления (объект класса LabelText) для пользователя, оповещающее о результате игры, а также поля ввода имени игрока, для последующей записи в таблицу рекордов. Новый рекорд заносится в текстовый файл, который позже считывается для отображения таблицы рекордов. После ввода имени пользователь выходит в игровое меню и имеет возможность, либо начать новую игру или же просмотреть таблицу рекордов. Если результат игры оказался одним из 5 лучших, то такой результат заносится в таблицу рекордов (объект класса RecordMenu). В таблице рекордов учитывается кол-во монет, набранных пользователем в ходе игровой сессии, затем метод Sort, класса RecordMenu, производит сортировку рекордов по убывании.

Подробная архитектура программы изображена на рисунке 1 и 2.

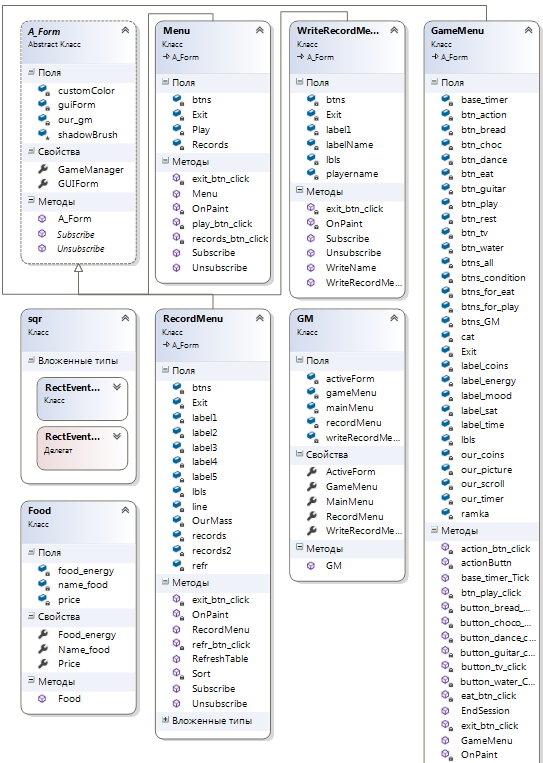


Рисунок 1 – Архитектура программы

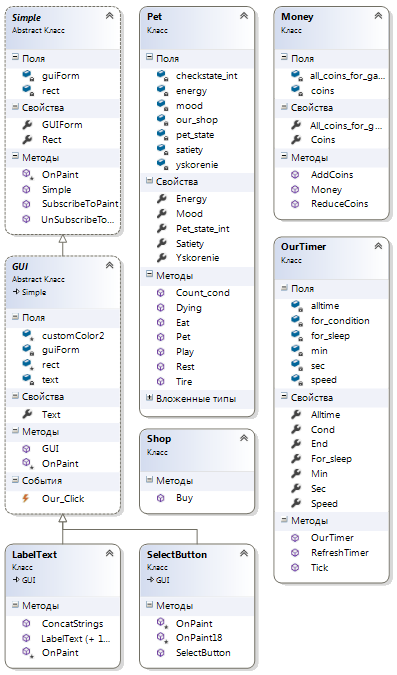


Рисунок 2 – Архитектура программы

**GameManager**

Класс GameManager осуществляет логику перехода между окнами игры.

Методы класса:

* public GameManager(Form1 guiForm) – Конструктор класса;

**AbstractForm**

Класс AbstractForm представляет абстрактный класс, который будет переопределяться при создании пользовательского окна.

Методы класса:

* public A\_Form(Form1 guiForm, GM gameManager) – Конструктор класса;
* public abstract void Subscribe( ) - Подписывает на события класса;
* public abstract void Unsubscribe( ) - Отписывает от событий класса;

**Menu**

Menu реализует игровые меню, состоящие из кнопок (компоненты класса SelectButton).

Методы класса:

* private void OnPaint(object sender, PaintEventArgs e) – Производит подписку на события данной формы;
* private void play\_btn\_click(object sender, MouseEventArgs e) – Обработка события нажатия на кнопку Play. Происходит начало игры.
* private void records\_btn\_click(object sender, MouseEventArgs e) - Обработка события нажатия на кнопку Records. Открывает таблицу результатов.
* private void exit\_btn\_click(object sender, MouseEventArgs e) – Обработка события нажатия на кнопку Exit. Выходит из игры.

**GameMenu**

GameMenu управляет игровым процессом.

Методы класса:

* private void OnPaint(object sender, PaintEventArgs e) – Производит подписку на события данной формы;
* public void base\_timer\_Tick(object sender, EventArgs e) – Каждую секунду производит изменения состояний питомца;
* public void RecordNotes() – Записывает данные сессии в файл;
* public void EndSession() – Завершает игру и обнуляет все состояния для последующей сессии;
* private void eat\_btn\_click(object sender, MouseEventArgs e) – Нажатие на кнопку кормления питомца;
* private void btn\_play\_click(object sender, MouseEventArgs e) – Нажатие на кнопку игры с питомцем;
* private void exit\_btn\_click(object sender, MouseEventArgs e) – Выход в главное меню;
* private void rest\_btn\_click(object sender, MouseEventArgs e) – Нажатие на кнопку отдыха;

**RecordMenu**

RecordMenu отображает таблицу результатов.

Методы класса:

* private void OnPaint(object sender, PaintEventArgs e) – Производит подписку на события данной формы;
* public void RefreshTable() – Обновляет данные таблицы рекордов;
* private void exit\_btn\_click(object sender, MouseEventArgs e) – Выход в главное меню;
* private void Sort(string input\_string) – Сортирует по убыванию данные, выбранные из файла;

**LabelText**

LabelText реализует создание текстового элемента

Метод класса:

* public override void OnPaint(object sender, PaintEventArgs e) – Рисует элемент.

**SelectButton**

SelectButton реализует кнопки в игровых меню.

Метод класса:

* public override void OnPaint(object sender, PaintEventArgs e) – Рисует кнопку.

**Food**

Класс Food содержит информацию о еде.

Методы класса:

* public Food(string type\_of\_food) – Конструктор класса;

**Shop**

Класс Shop производит покупку еды и забирает монеты.

Методы класса:

* public int Buy(string food\_name, Money our\_money) – Производит проверку на достаток монет и покупает еду;

**OurTimer**

OurTimer реализует таймер для данной игры, с возможностью ускорения и замедления игрового процесса.

Методы класса:

* public void Tick(Pet ourpet, OurTimer current\_speed) – Каждую секунду изменяет игровые параметры;
* public void RefreshTimer() – Обновляет таймер по окончанию игры для следующей сессии;
* public OurTimer() – Конструктор класса;

**Pet**

Класс Pet реализует создание питомца и контроль за его состоянием.

Методы класса:

* public void Eat(string type\_of\_food, Money our\_money) – Питомец питается, увеличивается параметр сытости;
* public void Play(string type\_of\_play, Money our\_money) – Питомец играет, увеличивается параментр настроения;
* public void Dying() – Питомец умирает;
* public void Count\_cond() – Производит проверку состояния питомца для последующей отрисовки;
* public void Tire() – Каждую секунду уменьшает параметры питомца;
* public void Rest() – Питомец отдыхает, увеличивает параметр энергии;

**Руководство пользователя**

Установка игры осуществляется копированием её программных файлов в каталог на жестком диске.

Загрузка игры осуществляется путём запуска исполняемого файла Tamagotchi.exe из директории программы на жестком диске.

В программе реализован графический интерфейс. При загрузке программы отображается главное меню программы со следующими пунктами (Рисунок 1):

1. Играть
2. Лучший результат
3. Выход

Навигация по меню осуществляется с помощью мыши. Для выбора пункта меню пользователю необходимо навести курсор на нужный пункт и нажать левую кнопку мыши.

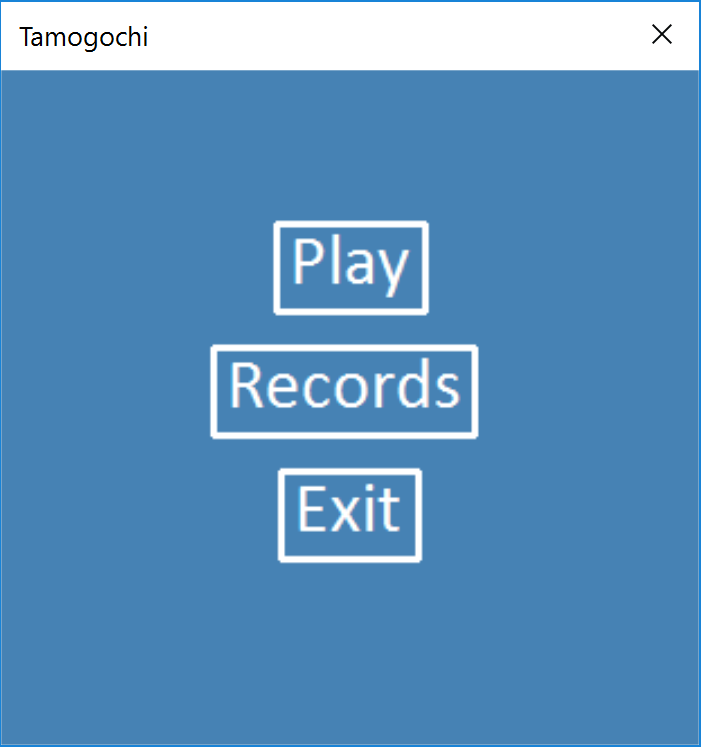


Рисунок 1. Главное меню

При нажатии на кнопку «Play» отображается игровое поле (Рисунок 2).

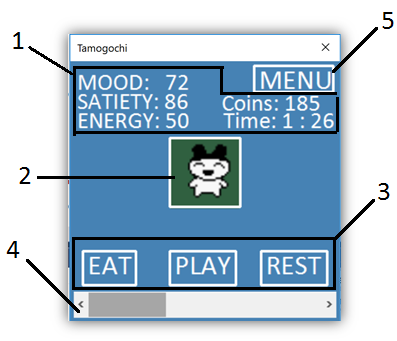


Рисунок 2. Игровое поле

На игровом поле находится пять элементов:

1. Панель информации о питомце и состоянии игры – Содержит всю информацию о питомце, такую как процент Сытости (Satiety), Настроение(Mood) и Энергию(Energy). Также на данной панели размещено кол-во Монет(Coins), которыми владеет питомец, и время данной игровой сессии(Time).
2. Индикатор питомца – Визуализирует состояние питомца в данный момент игрового времени.
3. Панель управления питомцем – Содержит кнопки Питаться(Eat), Играть(Play), Отдыхать(Rest), при нажатии которых пользователь управляет действиями питомца.
4. Панель регулировки времени – Изменяя положение ползунка пользователь регулирует игровое время, тем самым ускоряя или замедляя игровой процесс.
5. Кнопка выхода в главное меню – При нажатии пользователь возвращается в главное меню.

Правила игры: В игровое меню выводится питомец с личными характеристиками. Суть игры — следить за своим питомцем, кормить, ухаживать за ним.

В игре существует несколько различных состояний питомца (Рисунок 3).

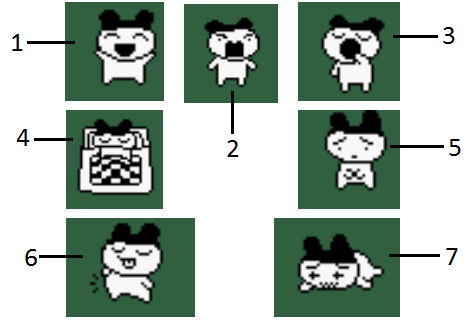


Рисунок 3. Состояния питомца

1. Счастливый – Все параметры больше 60%;
2. Злой – Параметр настроения меньше 40%;
3. Усталый – Параметр энергия меньше 40%;
4. Спящий – Питомец спит;
5. Голодный – Параметр сытости меньше 40%;
6. Сытый – Параметр сытости больше 40%;
7. Питомец умер – Параметр сытости равен 0%;

Основная потребность вашего питомца – Еда(Satiety). Чтобы покормить питомца, нужно кликнуть на кнопку **Action**и накормить питомца (выбрать вкладку Eat). От выбора вашей еды зависит число восстановленных процентов еды. Чем плотнее завтрак, тем дороже он стоит. На покупку нужно тратить монету(Coins). При нехватке монет, питомец не сможет приобрести еду.

В зависимости от текущего состояния питомца, требуется производить соответствующие действия. Так, когда питомец находится в состоянии 2 (Злой), требуется увеличивать параметр настроение(Mood), путём нажатия на клавишу играть (Play). В зависимости от того какой тип игры вы выберете, будет соответствовать кол-во процентов настроения, которые вы получите, а также кол-во монеток, которые вы заработаете, но также вы потратите больше энергии. При нажатии на кнопку спасть(Rest) питомец переходит в режим сна (состояние 4). Питомец полностью восстанавливает энергию по окончанию действия.

В ходе игры у питомца тратится параметр энергия(Energy). Для ее восполнения следует отдыхать. Осуществляется при нажатии на кнопку спасть(Rest), таким образом питомец переходит в режим сна (состояние 4). Питомец полностью восстанавливает энергию по окончанию действия.

Игра заканчивается, когда у питомца параметр сытости равен 0%. По окончанию игры пользователю необходимо ввести имя, под которым он будет записан в списке рекордов (Рисунок 4). Количество монеток, набранное пользователем в процессе игры, сохраняется и, если данный результат попадает в первую пятёрку лучших результатов, заносится в таблицу рекордов.

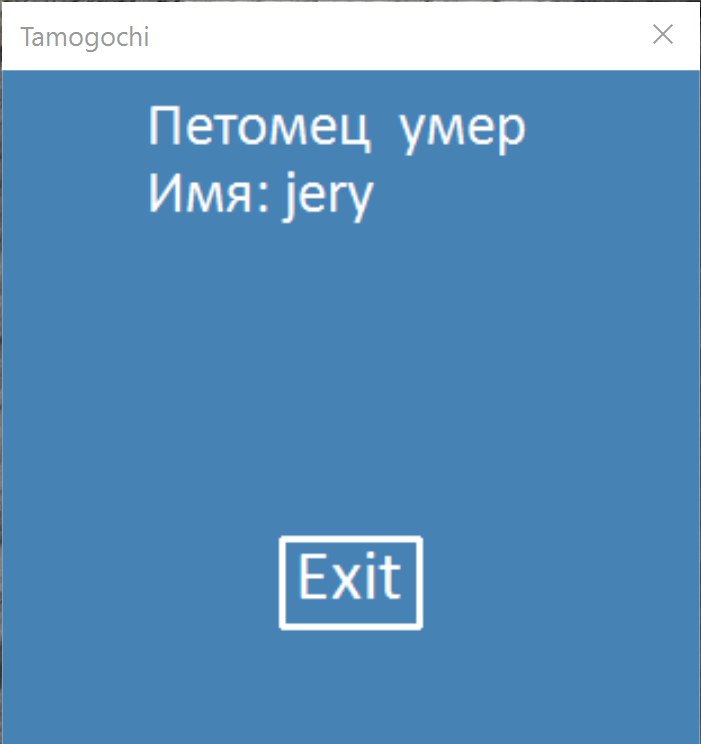


Рисунок 4. Конец игры

**При выборе пользователем пункта «**Records» открывается окно с таблицей рекордов и кнопкой ОК. Нажатие левой кнопки мыши на кнопке ОК возвращает в главное меню.

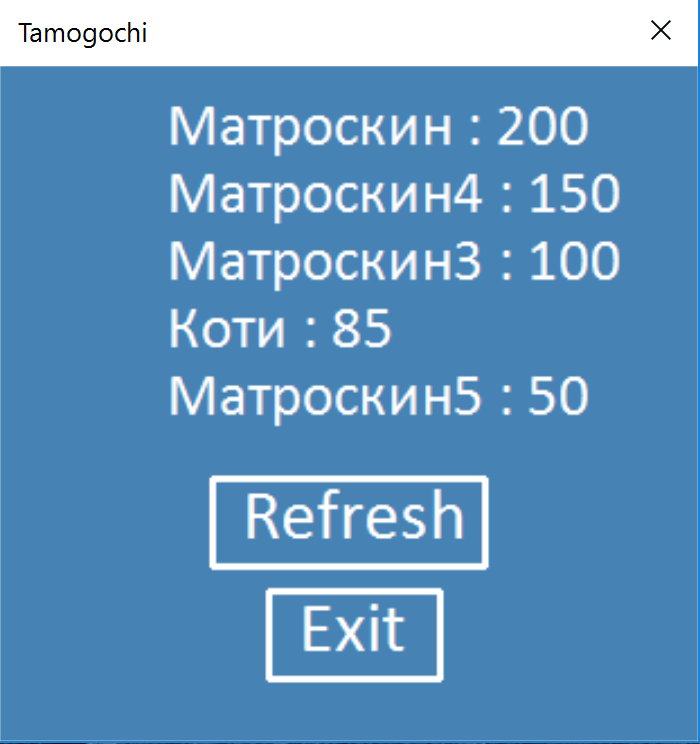


Рисунок 5. Меню «Рекорды»

При выборе пункта Exit происходит выход из программы.

**Приложение**

public abstract class A\_Form

{

private GM our\_gm;

private Form1 guiForm;

Color customColor;

protected SolidBrush shadowBrush;

public A\_Form(Form1 guiForm, GM gameManager)

{

this.our\_gm = gameManager;

this.guiForm = guiForm;

customColor = Color.FromArgb(255, (byte)70, (byte)130, (byte)180);

shadowBrush = new SolidBrush(customColor);

}

public GM GameManager

{

get { return our\_gm; }

}

public Form1 GUIForm

{

get { return guiForm; }

}

public abstract void Subscribe();

public abstract void Unsubscribe();

}

public class Menu : A\_Form

{

private GUI Play;

private GUI Records;

private GUI Exit;

private List<GUI> btns;

public Menu(Form1 guiForm, GM gameManager) : base(guiForm, gameManager)

{

Play = new SelectButton(GUIForm, new RectangleF(123.0f, 65.0f, 65.0f, 40.0f), "Play");

Records = new SelectButton(GUIForm, new RectangleF(95.0f, 120.0f, 115.0f, 40.0f), "Records");

Exit = new SelectButton(GUIForm, new RectangleF(125.0f, 175.0f, 60.0f, 40.0f), "Exit");

btns = new List<GUI>();

btns.Add(Play);

btns.Add(Records);

btns.Add(Exit);

guiForm.Invalidate();

}

public override void Subscribe()

{

GUIForm.Paint += OnPaint;

foreach (GUI btn in btns)

{

btn.SubscribeToPaint();

}

GUIForm.MouseClick += play\_btn\_click;

GUIForm.MouseClick += records\_btn\_click;

GUIForm.MouseClick += exit\_btn\_click;

GUIForm.Invalidate();

}

public override void Unsubscribe()

{

GUIForm.Paint -= OnPaint;

foreach (GUI btn in btns)

{

btn.UnSubscribeToPaint();

}

GUIForm.MouseClick -= play\_btn\_click;

GUIForm.MouseClick -= records\_btn\_click;

GUIForm.MouseClick -= exit\_btn\_click;

}

private void OnPaint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

Font drawFont = new Font("Arial", 20);

g.FillRectangle(shadowBrush, new Rectangle(new Point(), GUIForm.Size));

}

private void play\_btn\_click(object sender, MouseEventArgs e)

{

Unsubscribe();

if (Play.Rect.Contains(e.X, e.Y))

GameManager.ActiveForm = GameManager.GameMenu;

}

private void records\_btn\_click(object sender, MouseEventArgs e)

{

if (Records.Rect.Contains(e.X, e.Y))

GameManager.ActiveForm = GameManager.RecordMenu;

}

private void exit\_btn\_click(object sender, MouseEventArgs e)

{

if (Exit.Rect.Contains(e.X, e.Y))

{

GUIForm.Close();

Unsubscribe();

}

}

}

public class RecordMenu : A\_Form

{

private GUI Exit;

private GUI refr;

private List<GUI> btns;

private List<LabelText> lbls;

private List<OurTip> OurMass;

string[] records;

string[] records2;

string line;

LabelText label1;

LabelText label2;

LabelText label3;

LabelText label4;

LabelText label5;

public RecordMenu(Form1 guiForm, GM gameManager) : base(guiForm, gameManager)

{

RefreshTable();

Exit = new SelectButton(GUIForm, new RectangleF(120.0f, 230.0f, 75.0f, 40.0f), " Exit ");

refr = new SelectButton(GUIForm, new RectangleF(95.0f, 180.0f, 120.0f, 40.0f), " Refresh ");

guiForm.Invalidate();

}

public override void Subscribe()

{

GUIForm.Paint += OnPaint;

Exit.SubscribeToPaint();

refr.SubscribeToPaint();

GUIForm.MouseClick += exit\_btn\_click;

GUIForm.MouseClick += refr\_btn\_click;

GUIForm.Invalidate();

}

public override void Unsubscribe()

{

GUIForm.Paint -= OnPaint;

foreach (GUI btn in btns)

{

btn.UnSubscribeToPaint();

}

GUIForm.MouseClick -= exit\_btn\_click;

GUIForm.MouseClick -= refr\_btn\_click;

}

private void OnPaint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

Font drawFont = new Font("Arial", 20);

g.FillRectangle(shadowBrush, new Rectangle(new Point(), GUIForm.Size));

}

private void exit\_btn\_click(object sender, MouseEventArgs e)

{

if (Exit.Rect.Contains(e.X, e.Y))

{

GameManager.ActiveForm = GameManager.MainMenu;

Unsubscribe();

}

}

private void refr\_btn\_click(object sender, MouseEventArgs e)

{

if (refr.Rect.Contains(e.X, e.Y))

{

RefreshTable();

}

}

private void Sort(string input\_string)

{

records = input\_string.Split(new char[] { '@' });

for (int i = 0; i < records.Length - 1; i++)

{

records2 = records[i].Split(new char[] { ',' });

OurMass.Add(new OurTip {Name = records2[1], Coins = records2[0] });

}

string temp = "";

string temp2 = "";

for (int i = 0; i < OurMass.Count - 1; i ++)

{

for (int j = i + 1; j < OurMass.Count; j++)

{

if (Convert.ToInt64(OurMass[i].Coins) < Convert.ToInt64(OurMass[j].Coins))

{

temp = OurMass[i].Coins;

temp2 = OurMass[i].Name;

OurMass[i].Coins = OurMass[j].Coins;

OurMass[i].Name = OurMass[j].Name;

OurMass[j].Coins = temp;

OurMass[j].Name = temp2;

}

}

}

}

public class OurTip

{

public string Name { set; get; }

public string Coins { set; get; }

}

}

public class GameMenu : A\_Form

{

public GameMenu(Form1 guiForm, GM gameManager) : base(guiForm, gameManager)

{

cat = new Pet();

guiForm.Invalidate();

}

public void base\_timer\_Tick(object sender, EventArgs e)

{

GUIForm.Invalidate();

foreach (LabelText lbl in lbls)

{

lbl.UnSubscribeToPaint();

}

if (our\_timer.For\_sleep < 2)

{

cat.Pet\_state\_int = 6;

}

else cat.Pet\_state\_int = 5;

cat.Count\_cond();

if(our\_timer.Sec == 20)

{

cat.Yskorenie += 0.2; // игра быстрее идет к концу

}

our\_timer.Sec++;

if (our\_timer.Sec == 60)

{

our\_timer.Min += 1;

our\_timer.Sec -= 60;

}

our\_timer.Alltime = Convert.ToString(our\_timer.Min + " : " + our\_timer.Sec);

if (our\_timer.End != true)

{

label\_time = new LabelText(GUIForm, new RectangleF(172.0f, 52.0f, 40.0f, 20.0f), "Time: ", our\_timer.Alltime);

if (our\_timer.Sec % our\_timer.Speed == 0)

{

our\_timer.For\_sleep++;

cat.Tire();

}

}

foreach (LabelText lbl in lbls)

{

lbl.UnSubscribeToPaint();

}

}

public override void Subscribe()

{

GUIForm.Paint += OnPaint;

start\_the\_game();

foreach (GUI btn in btns\_GM)

{

btn.SubscribeToPaint();

}

GUIForm.MouseClick += btn\_play\_click;

GUIForm.MouseClick += exit\_btn\_click;

GUIForm.MouseClick += action\_btn\_click;

GUIForm.MouseClick += eat\_btn\_click;

GUIForm.MouseClick += rest\_btn\_click;

GUIForm.Invalidate();

}

public override void Unsubscribe()

{

GUIForm.Paint -= OnPaint;

foreach (GUI btn in btns\_all)

{

btn.UnSubscribeToPaint();

}

GUIForm.MouseClick -= play\_pet\_btn\_click;

GUIForm.MouseClick -= btn\_play\_click;

GUIForm.MouseClick -= exit\_btn\_click;

GUIForm.MouseClick -= button\_bread\_Click;

GUIForm.MouseClick -= button\_water\_Click;

GUIForm.MouseClick -= button\_choco\_Click;

GUIForm.MouseClick -= button\_guitar\_click;

GUIForm.MouseClick -= button\_tv\_click;

GUIForm.MouseClick -= button\_dance\_click;

GUIForm.MouseClick -= eat\_btn\_click;

GUIForm.MouseClick -= rest\_btn\_click;

base\_timer.Tick -= base\_timer\_Tick;

GUIForm.Load -= base\_timer\_Tick;

our\_timer.Sec = 0;

our\_scroll.Hide();

}

private void OnPaint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

Font drawFont = new Font("Arial", 20);

g.FillRectangle(shadowBrush, new Rectangle(new Point(), GUIForm.Size));

switch (cat.Pet\_state\_int)

{

case 1:

{

if (our\_timer.Cond % 2 == 0) g.DrawImage(Properties.Resources.angry1, 115, 90, 80, 80);

else g.DrawImage(Properties.Resources.angry2, 115, 90, 80, 80);

our\_timer.Cond += 1;

break;

}

case 2:

{

if (our\_timer.Cond % 2 == 0) g.DrawImage(Properties.Resources.tired1, 115, 90, 80, 80);

else g.DrawImage(Properties.Resources.tired2, 115, 90, 80, 80);

our\_timer.Cond += 1;

break;

}

case 3:

{

if (our\_timer.Cond % 2 == 0) g.DrawImage(Properties.Resources.death, 115, 90, 80, 80);

else g.DrawImage(Properties.Resources.death, 115, 90, 80, 80);

our\_timer.Cond += 1;

EndSession();

break;

}

case 4:

{

if (our\_timer.Cond % 2 == 0) g.DrawImage(Properties.Resources.full\_eat, 115, 90, 80, 80);

else g.DrawImage(Properties.Resources.full\_eat, 115, 90, 80, 80);

our\_timer.Cond += 1;

break;

}

case 5:

{

if (our\_timer.Cond % 2 == 0) g.DrawImage(Properties.Resources.happy1, 115, 90, 80, 80);

else g.DrawImage(Properties.Resources.happy2, 115, 90, 80, 80);

our\_timer.Cond += 1;

break;

}

case 6:

{

if(our\_timer.Cond % 2 == 0) g.DrawImage(Properties.Resources.sleep, 115, 90, 80, 80);

else g.DrawImage(Properties.Resources.sleep, 115, 90, 80, 80);

our\_timer.Cond += 1;

break;

}

case 8:

{

if (our\_timer.Cond % 2 == 0) g.DrawImage(Properties.Resources.want\_eat1, 115, 90, 80, 80);

else g.DrawImage(Properties.Resources.want\_eat2, 115, 90, 80, 80);

our\_timer.Cond += 1;

break;

}

}

}

private void actionButtn()

{

GUIForm.MouseClick += eat\_btn\_click;

GUIForm.MouseClick += btn\_play\_click;

GUIForm.MouseClick += rest\_btn\_click;

GUIForm.MouseClick += rest\_btn\_click;

foreach (GUI btn in btns\_for\_eat)

{

btn.UnSubscribeToPaint();

}

foreach (GUI btn in btns\_for\_play)

{

btn.UnSubscribeToPaint();

}

foreach (GUI btn in btns\_condition)

{

btn.UnSubscribeToPaint();

}

foreach (GUI btn in btns\_condition)

{

btn.SubscribeToPaint();

}

GUIForm.MouseClick -= button\_bread\_Click;

GUIForm.MouseClick -= button\_water\_Click;

GUIForm.MouseClick -= button\_choco\_Click;

GUIForm.MouseClick -= button\_guitar\_click;

GUIForm.MouseClick -= button\_tv\_click;

GUIForm.MouseClick -= button\_dance\_click;

GUIForm.MouseClick -= rest\_btn\_click;

}

public void RecordNotes()

{

string path = @"C:/Users/Pavel/Desktop/С#/Tamagochi(AlmostFinal)/Records.txt";

File.AppendAllText(path, "КотБезИмени" + "," + our\_coins.All\_coins\_for\_game + "@", Encoding.UTF8);

}

public void EndSession()

{

our\_timer.End = true;

if (our\_timer.Sec % 5 == 0)

{

base\_timer.Stop();

our\_timer.RefreshTimer();

string path = @"C:/Users/Pavel/Desktop/С#/Tamagochi(AlmostFinal)/Records.txt";

File.AppendAllText(path, our\_coins.All\_coins\_for\_game + ",", Encoding.UTF8);

Unsubscribe();

cat = new Pet();

GameManager.ActiveForm = GameManager.WriteRecordMenu;

}

}

private void action\_btn\_click(object sender, MouseEventArgs e)

{

if (btn\_action.Rect.Contains(e.X, e.Y))

{

GUIForm.MouseClick += eat\_btn\_click;

GUIForm.MouseClick += btn\_play\_click;

GUIForm.MouseClick += rest\_btn\_click;

GUIForm.MouseClick += rest\_btn\_click;

foreach (GUI btn in btns\_for\_eat)

{

btn.UnSubscribeToPaint();

}

foreach (GUI btn in btns\_for\_play)

{

btn.UnSubscribeToPaint();

}

foreach (GUI btn in btns\_condition)

{

btn.UnSubscribeToPaint();

}

foreach (GUI btn in btns\_condition)

{

btn.SubscribeToPaint();

}

GUIForm.MouseClick -= button\_bread\_Click;

GUIForm.MouseClick -= button\_water\_Click;

GUIForm.MouseClick -= button\_choco\_Click;

GUIForm.MouseClick -= button\_guitar\_click;

GUIForm.MouseClick -= button\_tv\_click;

GUIForm.MouseClick -= button\_dance\_click;

GUIForm.MouseClick -= rest\_btn\_click;

}

}

private void start\_the\_game()

{

our\_coins = new Money();

our\_timer.End = false;

GUIForm.MouseClick -= exit\_btn\_click;

foreach (GUI btn in btns\_condition)

{

btn.SubscribeToPaint();

}

base\_timer.Interval = 1000;

base\_timer.Start();

base\_timer.Tick += base\_timer\_Tick;

our\_scroll.Scroll += our\_scroll\_Scroll;

GUIForm.Load += base\_timer\_Tick;

GUIForm.Controls.Add(this.our\_scroll);

}

}

public class WriteRecordMenu: A\_Form

{

private GUI Exit;

private List<GUI> btns;

private List<LabelText> lbls;

private string playername = "Неизвестный";

LabelText label1;

LabelText labelName;

public WriteRecordMenu(Form1 guiForm, GM gameManager) :base(guiForm, gameManager)

{

lbls = new List<LabelText>();

lbls.Add(label1);

lbls.Add(labelName);

btns = new List<GUI>();

btns.Add(Exit);

GUIForm.KeyPress += WriteName;

guiForm.Invalidate();

}

private void exit\_btn\_click(object sender, MouseEventArgs e)

{

if (Exit.Rect.Contains(e.X, e.Y))

{

string path = @"C:/Users/Pavel/Desktop/С#/Tamagochi(AlmostFinal)/Records.txt";

File.AppendAllText(path, playername + "@", Encoding.UTF8);

GameManager.ActiveForm = GameManager.MainMenu;

Unsubscribe();

}

}

private void OnPaint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

Font drawFont = new Font("Arial", 20);

g.FillRectangle(shadowBrush, new Rectangle(new Point(), GUIForm.Size));

}

public void WriteName(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar >= 97 && e.KeyChar <= 122 && playername.Length < 10) ||

(e.KeyChar >= 48 && e.KeyChar <= 57 && playername.Length < 10))

{

playername += e.KeyChar.ToString();

}

if (playername.Length > 0 && e.KeyChar == (Char)Keys.Back)

{

playername = playername.Substring(0, playername.Length - 1);

}

GUIForm.Invalidate();

labelName.UnSubscribeToPaint();

labelName = new LabelText(GUIForm, new RectangleF(60.0f, 38.0f, 40.0f, 20.0f), "Имя: ", playername);

labelName.SubscribeToPaint();

if (e.KeyChar == (Char)Keys.Enter)

{

string path = @"C:/Users/Pavel/Desktop/С#/Tamagochi(AlmostFinal)/Records.txt";

File.AppendAllText(path, playername + "@", Encoding.UTF8);

GUIForm.KeyPress -= WriteName;

}

}

public class GM

{

private A\_Form activeForm;

private Menu mainMenu;

private GameMenu gameMenu;

private RecordMenu recordMenu;

private WriteRecordMenu writeRecordMenu;

public GM(Form1 guiForm)

{

mainMenu = new Menu(guiForm, this);

gameMenu = new GameMenu(guiForm, this);

recordMenu = new RecordMenu(guiForm, this);

writeRecordMenu = new WriteRecordMenu(guiForm, this);

ActiveForm = mainMenu;

}

public A\_Form ActiveForm

{

get { return activeForm; }

set

{

activeForm = value;

activeForm.Subscribe();

}

}

public WriteRecordMenu WriteRecordMenu

{

get { return writeRecordMenu; }

}

public Menu MainMenu

{

get { return mainMenu; }

}

public GameMenu GameMenu

{

get { return gameMenu; }

}

public RecordMenu RecordMenu

{

get { return recordMenu; }

}

}

class Square // для отрисовки и нажатий

{

public delegate void RectEventHandler(object sender, RectEventArgs e);

public class RectEventArgs

{

private RectangleF rect;

public RectEventArgs(RectangleF rect)

{

this.rect = rect;

}

public RectangleF Rect

{

get { return rect; }

set { rect = value; }

}

}

}

public abstract class Simple // абстрактный класс всех кнопок и меню

{

private RectangleF rect;

private Form1 guiForm;

public Simple(Form1 guiForm, RectangleF rect)

{

this.guiForm = guiForm;

this.rect = rect;

}

public RectangleF Rect

{

get { return rect; }

set { rect = value; }

}

public Form1 GUIForm

{

get { return guiForm; }

set { guiForm = value; }

}

public virtual void SubscribeToPaint()

{

guiForm.Paint += OnPaint;

}

public virtual void UnSubscribeToPaint()

{

guiForm.Paint -= OnPaint;

}

protected virtual void OnPaint(object sender, PaintEventArgs e) { }

}

public abstract class GUI : Simple

{

protected RectangleF rect;

private Form1 guiForm;

protected Color customColor2;

private string text;

public GUI(Form1 guiForm, RectangleF rect, string text) : base(guiForm,rect)

{

this.text = text;

customColor2 = Color.FromArgb(255, (byte)255, (byte)255, (byte)255);

}

public event MouseEventHandler Our\_Click; // для обработки события нажатия в нашем классе

protected override void OnPaint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

}

public string Text

{

get { return text; }

set { text = value; }

}

}

class LabelText : GUI

{

public LabelText(Form1 guiForm, RectangleF rect, string text) : base(guiForm, rect, text)

{

}

public LabelText(Form1 guiForm, RectangleF rect, string text1, string text2) : base(guiForm,rect,text1+text2)

{

}

protected override void OnPaint(object sender, PaintEventArgs e)

{

Font drawFont = new Font("Calibri", 20);

base.OnPaint(sender, e);

Graphics g = e.Graphics;

g.DrawString(Text, drawFont, new SolidBrush(customColor2), new PointF(Rect.X, Rect.Y));

}

public string ConcatStrings(string text1, string text2)

{

string onestring = "";

onestring = text1 + text2;

return onestring;

}

}

class SelectButton : GUI

{

public SelectButton(Form1 guiForm, RectangleF rect, string text) : base(guiForm, rect, text)

{

this.rect = rect;

}

protected override void OnPaint(object sender, PaintEventArgs e)

{

Font drawFont = new Font("Calibri", 24);

base.OnPaint(sender, e);

Graphics g = e.Graphics;

Pen pen = new Pen(Brushes.White, 3);

pen.LineJoin = LineJoin.Bevel;//задаем скошенные углы

pen.MiterLimit = 4;//задаем ограничение толщины скошенных углов

g.DrawRectangle(pen, new Rectangle(Convert.ToInt32(rect.X - 1), Convert.ToInt32(rect.Y + 3), Convert.ToInt32(rect.Width + 1), Convert.ToInt32(rect.Height - 1) ));

//полное заполнение заднего фона

g.DrawString(Text, drawFont, new SolidBrush(customColor2), new PointF(Rect.X, Rect.Y));

}

protected void OnPaint18(object sender, PaintEventArgs e)

{

Font drawFont = new Font("Arial", 18);

base.OnPaint(sender, e);

Graphics g = e.Graphics;

g.DrawString(Text, drawFont, new SolidBrush(Color.White), new PointF(Rect.X, Rect.Y));

}

}

public class Money

{

private int coins;

private int all\_coins\_for\_game;

public Money()

{

coins = 300;

all\_coins\_for\_game = 0;

}

public int Coins

public int All\_coins\_for\_game

public void AddCoins(int number\_of\_coins)

{

coins += number\_of\_coins;

all\_coins\_for\_game += number\_of\_coins;

}

public void ReduceCoins(string name\_food)

{

if (coins > 0)

{

Food food = new Food(name\_food);

if (food.Name\_food == "bread")

{

if (coins >= 30) coins -= 30;

}

if (food.Name\_food == "water")

{

if (coins >= 20) coins -= 20;

}

if (food.Name\_food == "choco")

{

if (coins >= 10) coins -= 10;

}

}

}

}

public class Shop

{

public int Buy(string food\_name, Money our\_money)

{

Food type\_our\_food = new Food(food\_name);

if (our\_money.Coins >= type\_our\_food.Price)

{

return type\_our\_food.Food\_energy;

}

return 0;

}

}

public class Food

{

private int food\_energy;

private int price;

private string name\_food;

public int Food\_energy

{

get { return food\_energy; }

set { food\_energy = value; }

}

public int Price

{

get { return price; }

set { price = value; }

}

public string Name\_food

{

get { return name\_food; }

}

public Food(string type\_of\_food)

{

name\_food = type\_of\_food;

if (type\_of\_food == "bread")

{

food\_energy = 30;

price = 30;

}

if (type\_of\_food == "choco")

{

food\_energy = 10;

price = 10;

}

if (type\_of\_food == "water")

{

food\_energy = 20;

price = 20;

}

}

}

public class Pet

{

Shop our\_shop = new Shop();

private double satiety;

private double energy;

private double mood;

private int checkstate\_int;

private int pet\_state = 5;

private double yskorenie = 1;

public enum check\_pet\_state

{

angry = 1,

bad\_mood,

death,

full\_eat,

happy,

sleep,

tired,

hungry

}

public Pet()

{

satiety = 100;

energy = 100;

mood = 100;

}

public void Eat(string type\_of\_food, Money our\_money)

{

satiety += our\_shop.Buy(type\_of\_food, our\_money);

if (satiety > 100) satiety = 100;

}

public void Rest()

{

energy += 90;

if (energy > 100) energy = 100;

}

public void Play(string type\_of\_play, Money our\_money)

{

if (type\_of\_play == "Guitar")

{

mood += 10;

our\_money.All\_coins\_for\_game += 5;

our\_money.Coins += 5;

if (mood > 100) mood = 100;

}

if (type\_of\_play == "Video")

{

mood += 20;

our\_money.All\_coins\_for\_game += 10;

our\_money.Coins += 10;

if (mood > 100) mood = 100;

}

if (type\_of\_play == "Dance")

{

mood += 30;

our\_money.All\_coins\_for\_game += 15;

our\_money.Coins += 15;

if (mood > 100) mood = 100;

}

}

public void Count\_cond()

{

if (checkstate\_int == 6)

{

checkstate\_int = 6;

Console.WriteLine("sleep");

}

else

{

if ( satiety == 0 )

{

Console.WriteLine("death");

checkstate\_int = 3;

Dying();

}

if (satiety >= 60)

{

checkstate\_int = 4;

Console.WriteLine("full\_eat");

}

if (mood >= 60 && satiety !=0)

{

checkstate\_int = 5;

Console.WriteLine("happy");

}

if (energy <= 30 && energy > 0)

{

checkstate\_int = 2;

Console.WriteLine("tired");

}

if (mood <= 30 && satiety != 0)

{

checkstate\_int = 1;

Console.WriteLine("angry");

}

if(satiety <= 30 && satiety > 0)

{

checkstate\_int = 8;

Console.WriteLine("want eat");

}

}

}

public void Tire()

{

if (satiety >= 1) satiety -= 10 \* yskorenie;

if (satiety < 0) satiety = 0;

if (energy >= 1) energy -= 10 \* yskorenie;

if (energy < 0) energy = 0;

if (mood >= 1) mood -= 10 \* yskorenie;

if (mood < 0) mood = 0;

}

}