**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ЯРОСЛАВСКОЙ ОБЛАСТИ**

государственное профессиональное образовательное учреждение   
Ярославской области

Рыбинский полиграфический колледж

**КУРСОВОЙ ПРОЕКТ**

|  |  |
| --- | --- |
| Разработка класса визуального компонента-игры «Подбери пару» | |
|  | |
| по дисциплине | Объектно-ориентированное программирование |
| КП.0902.06.000000.00 ПЗ | |

Пояснительная записка

|  |
| --- |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент группы | 3-ИС-2 |  |  |  | Киселев В С |
|  | *(Код учебной группы)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |
| Руководитель | преподаватель |  |  |  | Смирнов В Б |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *( И.О.Фамилия)* |
| Нормоконтроль | преподаватель |  |  |  | Смирнов В Б |
|  | *(Должность, звание)* |  | *(Подпись, дата)* |  | *(И.О.Фамилия)* |

г. Рыбинск

2022

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc126154694)

[1 Исследовательский раздел 5](#_Toc126154695)

[2 Конструкторский раздел 7](#_Toc126154696)

[2.1 Математическая модель 7](#_Toc126154697)

[2.2 Разработка структуры компонента 8](#_Toc126154698)

[2.3 Разработка графического интерфейса компонента 10](#_Toc126154699)

[2.4 Описание входных и выходных данных компонента 11](#_Toc126154700)

[2.5 Разработка алгоритмов компонента 13](#_Toc126154701)

[3 Технологический раздел 19](#_Toc126154702)

[3.1 Отладка компонента 19](#_Toc126154703)

[3.2 Инструкция по установке компонента 20](#_Toc126154704)

[3.3 Инструкция по эксплуатации компонента 24](#_Toc126154705)

[4 Раздел охраны труда 26](#_Toc126154706)

[Заключение 28](#_Toc126154707)

[Список используемых источников 29](#_Toc126154708)

[Приложение А 30](#_Toc126154709)

Введение

Использование компьютерных технологий в современных разработках играет главную роль. Обработка информации при помощи компьютеров позволяет максимально быстро получить нужный результат. Именно поэтому появилась потребность в использовании специальных языков взаимодействия с вычислительной машиной.

С начала создания компьютеров были разработаны очень сложные для понимания языки машинного уровня, которые взаимодействовали напрямую с памятью устройства. В дальнейшем для более простого и быстрого написания кода, были разработаны языки программирования, которые существенно сократили временные затраты на разработку программного обеспечения.

Для того, чтобы уменьшить повторение кода, были разработаны объектно-ориентированные языки программирования, что помогло разработчикам соединить понимание кода с визуальным компонентом.

В информационном обществе выделяют три этапа развития языков.

Развитие языков началось с низкоуровневого программирования. Яркими примерами служат Assembler и C. В данном этапе развития команды обращались напрямую к процессору компьютера, за что и получили свое название. Минусами данного уровня являются наличие минимальных команд и ограниченный синтаксис ввода данных.

Следующим в ветке развития стали алгоритмические или функциональные языки. Одним из таких можно назвать Pascal. Данный язык состоит из набора операторов, что дает осмысленность написанного кода другому человеку.

С развитием общества потребность в создании прогрессирующего языка не заставила себя долго ждать – им стал объектно-ориентированный язык программирования. Он позволял ввод смысловых конструкций, описание структуры данных и их операции, что упрощает написание программных задач и облегчает понимание сути программы для человека. К ним можно отнести C#.

Целью курсового проекта является написание класса визуального компонента-игры «Подбери пару». Актуальность проекта заключается в первую очередь в его полезности для развития мелкой моторики, обучения логики мышления и запоминанию мелких деталей для детей дошкольного и младшего школьного возрастов. Прикладным программистам, начинающим разработчикам или студентам данный компонент поможет в понимании логики написания программного кода, просто ради развлечения или для улучшений своих разработок, посредством внедрения компонента в проекты и изменения базовых данных на свое усмотрение.

1 Исследовательский раздел

В ходе подготовки к разработке визуального компонента были рассмотрены две среды программирования:

* Microsoft Visual Studio 2019;
* Borland Delphi 2007.

Рассмотрим функционал данных сред.

Microsoft Visual Studio 2019 – это условно-бесплатный продукт, разработанный компанией Microsoft. Данная среда легко подойдет для начинающих программистов, поскольку сами разработчики Microsoft ведут обучающий блог о языке программирования C#. Эта среда позволяет разрабатывать пользовательский интерфейс и устанавливать сторонние библиотеки и дает подсказки при написании кода. Также она позволяет отслеживать найденные ошибки и диагностировать поведение созданной программы. Минусами данного продукта можно считать зависимость от операционной системы и комплектующих компьютера[1].

Borland Delphi 2007 – это платный продукт, разработанный компанией Borland. Достоинствами данной среды является скорость разработки компонентов, принцип открытости и универсальность. Минусами можно считать отсутствие перегрузки операторов и объектной модели, малое число параметров для оптимизации кода, большие размеры exe-файлов, а также отсутствие подсказок при написании кода. Но при этом он подойдет разработчикам, поскольку синтаксис написания кода довольно прост и не зависит от комплектующих компьютера[2].

После тщательного выявления достоинств и недостатков данных сред программирования, была выбрана среда программирования Microsoft Visual Studio 2019, так как она подходит для первых движений в сфере программирования, а также отлично подойдет для реализации проекта.

Чтобы удостовериться в актуальности реализации проекта был организован поиск аналоговых работ. В конечном итоге было найдено мало реализованных проектов, но у всех были одинаковые минусы – они были написаны при помощи WinForm.

В конечном итоге, при анализе найденных аналогов, разрабатываемый программный продукт будет обладать главным достоинством – он будет реализован отдельным компонентом. Его можно будет с легкостью добавлять в другие проекты, при желании изменять в нем данные, а также он будет находиться в свободном доступе.

Для разработки компонента был выбран родительский класс – Control. Данный класс обладает необходимым функционалом для разработки проекта: механизмы взаимодействия пользователя с компонентом при помощи манипулятора типа «мышь» и клавиатурой, механизмы рисования.

Основным процессом данного компонента будет являться сравнение изображений на открытых карточках для составления пары, что приводит в действие другие процессы: переворот и удаление.

2 Конструкторский раздел

2.1 Математическая модель

Как и все игры, данная разработка будет состоять из ряда правил для её прохождения.

Игра «Подбери пару» будет придерживаться одного правила – нужно найти все одинаковые изображения. Это игра будет одиночной.

В текущей разработке планируется создать 3 уровня сложности игры. Сложность будет заключаться в том, что при завершении уровня будет увеличиваться количество карточек на игровом поле.

Вначале игрок должен будет запустить механизм создания новой игры. Каждый раз игра будет начинаться с начального уровня – самый легкий, состоящий из 4 карточек по 2 пары. После прохождения предыдущего уровня игрок будет попадать на следующий уровень – состоящий из 6 карточек по 3 пары. И, наконец, при завершении 2 уровня, игрок будет попадать на последний уровень – сложный, состоящий из 8 карточек по 4 пар.

У игрока будет всего 2 попытки на нахождение пар. Попытки будут считаться посредством нажатия на карточку. При выборе элементов с разным изображением, обе карточки переворачиваются и попытки аннулируются, начинается выбор с самого начала. И так далее до завершения уровня. Затем будет создаваться новое поле.

Игра заканчивается, когда игрок найдет все пары и игровое поле последнего уровня будет полностью пустым. При завершении игры, будет выводиться сообщение о победе.

2.2 Разработка структуры компонента

Чёрная сфера - используется для обозначения работы системы, внутреннее устройство которой неизвестно. В данную систему поступают входные данные, а выходят выходные данные. Черная сфера представлена на рисунке 2.1.



Рисунок 2.1 – Черная сфера

Представим разрабатываемый класс в виде черной сферы. В компоненте будут присутствовать такие входные данные, как изображения карточек, управляющее воздействие пользователя и размеры. На выходе из компонента будет поступать итоговое изображение уровня на экране. Управляющее воздействие будет происходить с помощью манипулятора типа «мышь». Черная сфера с входными и выходными данными представлена на рисунке 2.2.



Рисунок 2.2 – Черная сфера с данными

Проведем первый этап декомпозиции для выявления структуры компонента. Размеры компонента и его элементов будут находиться в процентном соотношении. Обработка нажатия кнопки манипулятора «мышь» будет осуществляться событием, реагирующим на нажатие и изменяя текущее состояние элемента изображения. Результат первого этапа декомпозиции представлен на рисунке 2.3.



Рисунок 2.3 – Результат первого этапа декомпозиции

На первом этапе декомпозиции были получены основные структуры данных, необходимые для работы процессов компонента и хранения информации. Для получения более точных структур данных и процессов компонента проведем второй этап декомпозиции.

Управляющее воздействие пользователя будет влиять на следующие методы: изменение состояния карточки, старт и остановка игры. Метод старта игры изменит состояние игры и вызовет метод установки стартовых значений первого уровня. Метод остановки изменит состояние. При остановке таймера произойдет событие записи лучшего счета и обновление экрана. Результат второго этапа декомпозиции представлен на рисунке 2.4 и на плакате КП.0902.06.000000.01 ПЛ.



Рисунок 2.4 – Результат второго этапа декомпозиции

2.3 Разработка графического интерфейса компонента

Управление работой компонента будет происходить при помощи манипулятора типа «мышь». Поскольку данный компонент будет разрабатываться под операционные систему Windows, на форме компонент будет принимать прямоугольную форму.

Компонент будет выглядеть в виде поля из определенного количества карточек. Количество карточек, их расположение и размеры будут зависеть от уровня сложности. Расстояние между соседними элементами будет равняться половине ширины карточки для того, чтобы не было случайного нажатия на соседнюю карточку. Так, например, на первом уровне будет расположено 4 карточки в виде квадрата. На втором уровне карточки будут расположены в центре в количестве 6 штук, в виде прямоугольника. На третьем уровне карточек будет находиться 8 штук, и они будут расположены в виде прямоугольника.

2.4 Описание входных и выходных данных компонента

В подразделе 2.2 была получена структурная схема, из которой следует необходимость присутствия определенных переменных и событий компонента. Все свойства класса приведены в таблице 2.1.

Таблица 2.1 – Свойства класса

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя свойства | Тип данных | Тип доступа | Метод доступа на чтение | Метод доступа на запись |
| Score(Счет) | Int | чтение/запись | Вернёт значение переменной \_score | Используется механизм обновления значения переменной \_score, вызов события RecordScore |
| IsCoup(Проверка на схожесть) | Int | чтение/запись | Вернёт значение переменной \_left | Используется механизм обновления значения переменной  \_ left для инициализации количества оставшихся карточек |
| GameStatus(Статус игры) | Bool | чтение | Вернёт значение переменной \_gameStatus |  |

События класса представлены в таблице 2.2.

Таблица 2.2. – События класса

|  |  |  |  |
| --- | --- | --- | --- |
| Имя события | Тип данных | Тип доступа | Описание события |
| RecordScore(Запись счета) | EventHandler | добавление/удаление | Событие изменения лучшего количества очков. Будет срабатывать при завершении игры. |

2.5 Разработка алгоритмов компонента

В результатах подраздела 2.2, получена структурная схема компонента, из которой следует необходимость присутствия тех или иных методов для полноценной работы компонента. В соответствие с этой структурной схемой в данном разделе будет рассмотрена разработка алгоритмов всех методов.

GenerateCoup – метод отвечающий за размещение карточек на форме. В данном методе выбирается случайное изображение карточек в зависимости от их количества. Блок- схема алгоритма метода Random приведена на рисунке 2.3



Рисунок 2.3 – Алгоритм метода GenerateCoup

StartProcess – метод вызывающийся для старта игры. Он запускает игру и переводит фокус на компонент. Блок- схема алгоритма метода StartProcess приведена на рисунке 2.4



Рисунок 2.4 – Алгоритм метода StartProcess

EndProcess– метод вызывающийся при завершении игры. Он очищает форму, меняет значение флага состояния игры. В конце вызывается метод Invalidate который обновляет экран. Блок- схема алгоритма метода EndProcess приведена на рисунке 2.5



Рисунок 2.5 – Алгоритм метода EndProcess

AllClear– метод отвечающий за очищение формы. Входные и выходные параметры отсутствуют. Блок- схема алгоритма метода AllClear приведена на рисунке 2.6



Рисунок 2.6 – Алгоритм метода AllClear

OnPaint – метод отвечающий за прорисовку в компоненте. Входные параметры e типа PaintEventArgs, выходные параметры отсутствуют. Блок-схема алгоритма метода OnPaint приведена на рисунке 2.7.



Рисунок 2.7 – Алгоритм метода OnPaint

NextLvl– метод отвечающий за переход на следующий уровень. Входные и выходные параметры отсутствуют. Блок- схема алгоритма метода NextLvl приведена на рисунке 2.8



Рисунок 2.8 – Алгоритм метода NextLvl

IsCoup– метод отвечающий за проверку одинаковых карточек. Входные и выходные параметры отсутствуют. Блок- схема алгоритма метода IsCoup приведена на рисунке 2.9



Рисунок 2.9 – Алгоритм метода IsCoup

3 Технологический раздел

3.1 Отладка компонента

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки, информация из работы [3]. В связи с тем, что почти невозможно составить реальную программу без ошибок, и почти невозможно для достаточно сложной программы быстро найти и устранить все имеющиеся в ней ошибки.

Тестирование будет происходить через тест кейсы. Тест кейс – это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части. При передаче тестировщику тест-кейсов, он должен пройтись по всем его пунктам и выполнить описанные действия, которые должны привести к определенным результатам информации из работы[4]. Тест-кейс для свойств представлен в таблице 3.1.

Таблица 3.1 – Тест-кейс для свойств

|  |  |  |
| --- | --- | --- |
| Имя свойства | Управляющее воздействие | Результат воздействия |
| Score | Изменить значение свойства путём нахождения пары | Меняется значение переменной \_score и срабатывает событие RecordScore |

Тест-кейс для событий представлен в таблице 3.2.

Таблица 3.2 – Тест-кейс для событий

|  |  |  |
| --- | --- | --- |
| Имя события | Управляющее воздействие | Результат воздействия |
| RecordScore | Найти на игровом поле найти пару (две одинаковые карточки) | Счётчик очков Score увеличится на 2 единицы |

Тест-кейс для методов представлен в таблице 3.3.

Таблица 3.3 – Тест-кейс для методов

|  |  |  |
| --- | --- | --- |
| Имя метода | Управляющее воздействие | Результат воздействия |
| StartProcess | Программно вызвать метод | Генерация уровня, начало игры |
| EndProcess | Дойти до конца игры | Вывод сообщения об окончании игры |
| onMouseClick | Нажать кнопкой мыши по карточке | Появление изображения на нажатой карточке |
| OnPaint | Запустить игру | Прорисовка уровня  Вывод сообщения в зависимости от результата игры (победа или проигрыш) |
| SetBoundsCore | Изменение размеров компонента | Изменение ширины и высоты компонента |
| AllClear | Завершить игру или пройти уровень | Очистка игровой зоны |
| NextLvl | Пройти уровень | Вывод сообщения о загрузке уровня |
| IsCoup | Найти 2 карточки | Открытие карточек на 2 секунды, далее их закрытие или удаление (если они одинаковые) |
| GenerateCoup | Начать игру/перейти на следующий уровень | Сгенерированные случайные изображения |

3.2 Инструкция по установке компонента

Для установки компонента в проект необходимо в обозревателе решений щелчком правой кнопки мыши на ветке «Ссылки» в структуре текущего проекта вызвать контекстное меню. В появившемся меню необходимо выбрать команду «Добавить ссылку», как показано на рисунке 3.1

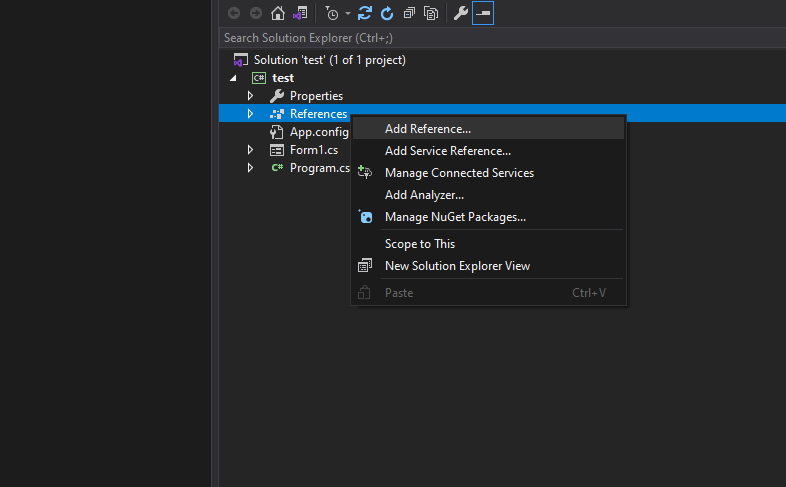


Рисунок 3.1 – Контекстное меню

После описанных действий на экране должно появиться окно Менеджера ссылок. Его внешний вид приведен на рисунке 3.2. В нем, с левой стороны, необходимо выделить раздел настроек «Обзор».

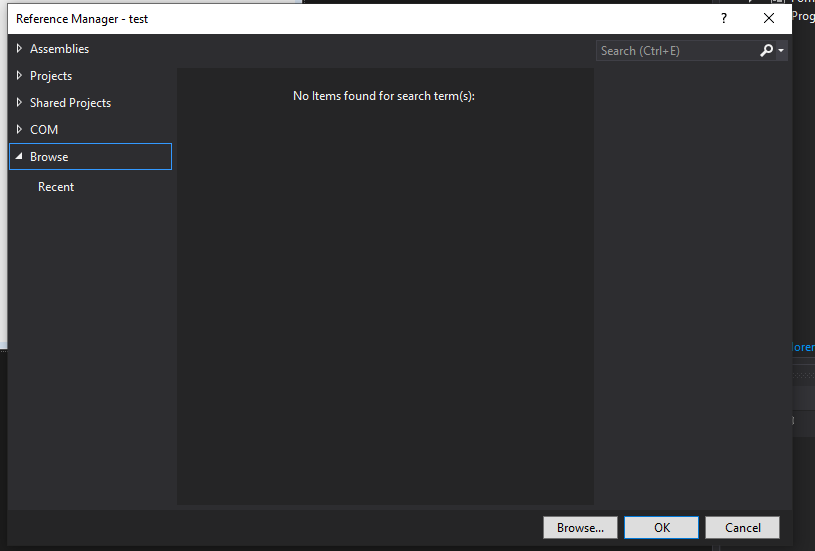
~~~~

Рисунок 3.2 – Менеджер ссылок

В появившемся на экране окне, предназначенном для выбора файлов, представленном на рисунке 3.3 требуется найти и выбрать файловый пакет с компонентом.

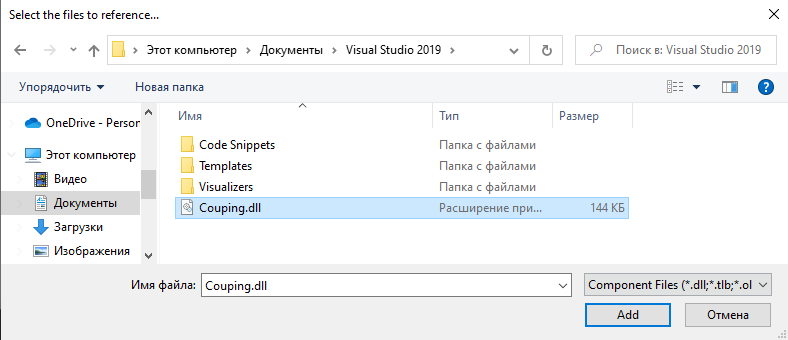


Рисунок 3.3 – Файловый пакет в проводнике

После выбора файлового пакета в панели элементов необходимо выбрать новую вкладку, и на вкладке щелчком правой кнопки мыши вызвать контекстное меню. В меню следует выбрать команду «Выбрать элементы», как показано на рисунке 3.4.

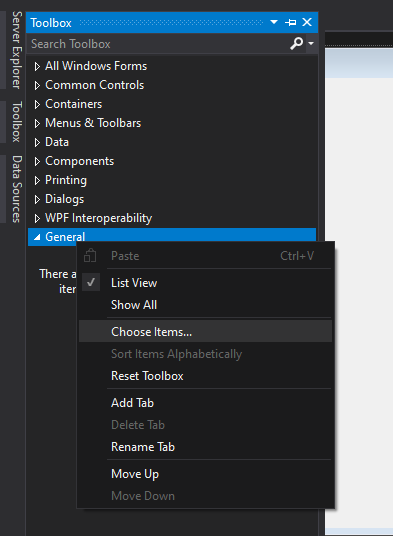


Рисунок 3.4 – Панель элементов

В появившемся окне «Выбор элементов панели элементов», в правом углу необходимо нажать на кнопку «Обзор» как показано на рисунке 3.5

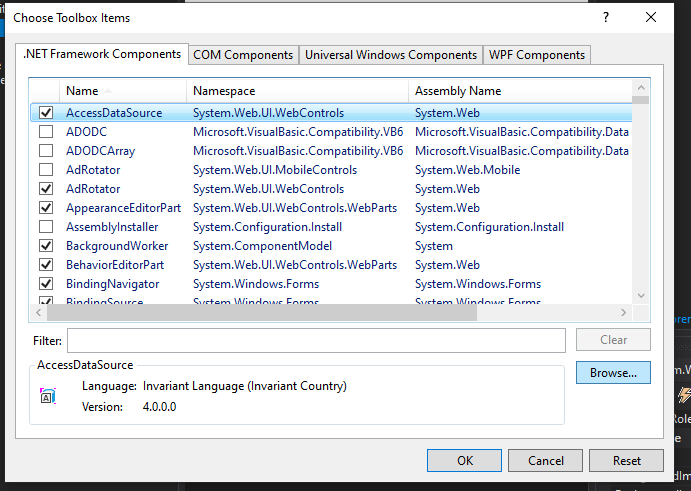


Рисунок 3.5 – Окно «Выбор элементов панели элементов»

Далее в появившемся окне проводника необходимо найти файловый пакет с компонентом и выбрать его, как показано на рисунке 3.6.

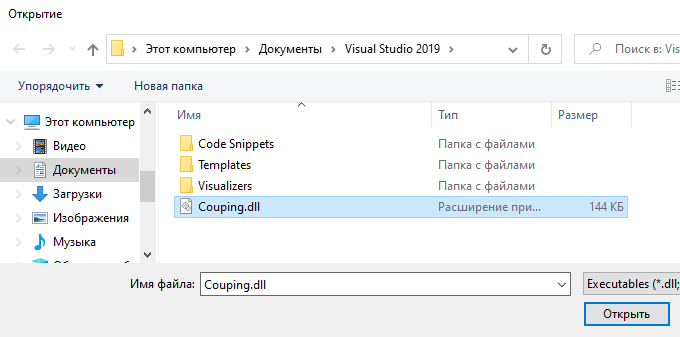


Рисунок 3.6 – Файловый пакет в проводнике

После завершения всех выше перечисленных действий, игровой компонент будет доступен для включения в прикладные программы через панель элементов.

Чтобы разместить компонент на форме приложения, требуется в панели элементов выбрать вкладку «New». В этой вкладке необходимо выделить компонент и переместить его на форму. Результат представлен на рисунке 3.7

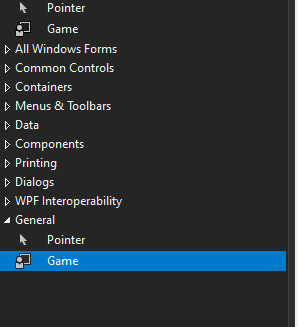


Рисунок 3.7 – Панель элементов

3.3 Инструкция по эксплуатации компонента

Для удобства вся инструкция по эксплуатации визуального компонента сведена в табличный вид. Все свойства компонента перечислены в таблице 4.1

Таблица 4.1 – Свойства компонента

|  |  |  |  |
| --- | --- | --- | --- |
| Имя свойства | Тип данных | Тип доступа | Назначение |
| Score (Счёт игры) | Int | чтение | Свойство необходимо для отображения текущего счёта |
| GameStatus (Статус игры) | Bool | чтение | Свойство необходимо для отображения текущего статуса игры (игра идет или закончена) |

События компонента представлены в таблице 4.2.

Таблица 4.2 – События компонента

|  |  |
| --- | --- |
| Имя события | Назначение |
| RecordScore | Событие срабатывает при изменении счёта |

Публичные методы компонента представлены в таблице 4.3.

Таблица 4.3 – Публичные методы компонента

|  |  |  |
| --- | --- | --- |
| Имя события | Входные и выходные параметры | Назначение |
| EndProcess | Нет входных и выходных параметров | Метод служит для принудительного завершения игры |
| StartProcess | Нет входных и выходных параметров | Метод служит для старта игры |

4 Раздел охраны труда

Охрана труда – это целая система законодательных и нормативно-правовых актов, технических, гигиенических, лечебно-профилактических мероприятий и средств, которые обеспечивают безопасность, сохранение здоровья и работоспособности человека в процессе труда. В наши дни труд стал более интенсивным и требует огромных затрат умственной, эмоциональной и физической нагрузок.

На рабочем месте программист осуществляет трудовую деятельность и проводит большую часть рабочего времени. Правильная организация рабочего места программиста повышает производительность труда от 8 до 20%. Следуя рекомендациям ГОСТ 12.2.032-78, необходимо организовать рабочее место таким образом, чтобы взаимное расположение всех его элементов соответствовало физическим и психологическим требованиям. Главные элементы рабочего места программиста – это письменный стол и кресло. Рабочее место организуется в соответствии с ГОСТ 12.2.032-78, информация из работы [5].

Площадь рабочего места с компьютером с жидкокристаллическим или плазменным экраном должна быть не менее 4,5 кв. м, а расстояние между столами с мониторами (от тыла одного монитора до экрана другого) не менее 2 м. Монитор должен располагаться на расстоянии 50-70 см от глаз программиста. Параметры рабочего стола сотрудника: возможность регулировки высоты рабочего стола, или точная высота — 72,5 см, ширина — 80, 100, 120 или 140 см, глубина рабочего стола 80 или 100 см, высота и ширина пространства под столешницей (для ног) – не менее 50 см, глубина на уровне колен не менее 45 см, а на уровне вытянутых ног не менее 65 см.

Правильное освещение рабочего места – это очень важный момент в трудовой деятельности человека, влияющий на эффективность труда, при этом такой момент предупреждает травматизм и профессиональные заболевания. При недостаточном освещении приходится напрягать зрение, при этом ослабляется внимание и это приводит к наступлению преждевременной утомленности. Слишком яркое освещение тоже плохо, так как оно вызывает ослепление, раздражение и резь в глазах. При искусственном освещении, источниками света служат два вида ламп: лампы накаливания и люминесцентные.

Известно, что шум ухудшает условия труда и оказывает вредное воздействие на организм человека. Согласно ГОСТ 12.1.003-88 «Шум для помещений расчетчиков и программистов, уровни шума не должны превышать соответственно: 71, 61, 54, 49, 45, 42, 40, 38 дБ», информация из работы [6].

При работе компьютерной техники выделяется много тепла, что может привести к пожароопасной ситуации. Источниками зажигания так же могут служить приборы, применяемые для технического обслуживания, устройства электропитания, кондиционеры воздуха. Серьёзную опасность представляют различные электроизоляционные материалы, используемые для защиты от механических воздействий отдельных радиодеталей. В связи с этим, участки, на которых используется компьютерная техника, по пожарной опасности относятся к категории пожароопасных “В”. При пожаре люди должны покинуть помещение в течение минимального времени. В помещениях с компьютерной техникой, недопустимо применение воды и пены ввиду опасности повреждения или полного выхода из строя дорогостоящего электронного оборудования. Для тушения пожаров необходимо применять углекислотные и порошковые огнетушители, которые обладают высокой скоростью тушения, большим временем действия, возможностью тушения электроустановок, высокой эффективностью борьбы с огнем. Воду разрешено применять только во вспомогательных помещениях, информация из работы [7].

Заключение

По итогу работы был разработан компонент игры «Подбери пару». Компонент является адаптивным и представляет собой поле из различного количества карточек. В зависимости от результата игры в окне будет появляться сообщение, сигнализирующее о том, что игра была завершена с определенным результатом. Если игрок прошел все уровни, то ему покажут сообщение об успешном прохождении игры.

Главным достоинством разработанного компонента можно считать отсутствие таймера, которая заставляла бы пользователя быстрее находить пары.

У компонента есть перспективы развития в плане визуализации какой-либо анимации в сопровождении со звуковыми эффектами, а также в реализации системы уровней с разным расположением карточек на уровне.

Разработанный игровой класс можно использовать в развлекательных целях, чтобы сократить время в ожидании чего-либо или просто развлечь себя. Так же компонент можно использовать в учебных целях. Компонент может проиллюстрировать различные методы рисования, объектно-ориентированный подход, работу с таймером и потоками. В компоненте используются различного рода алгоритмы, на которые можно опираться при разработке других компонентов.

Список используемых источников

1. <https://ruprogi.ru/software/visual-studio>
2. <https://blog.skillfactory.ru/glossary/delphi/>
3. <https://studopedia.ru/22_29871_neobhodimost-otladki-programmnogo-produkta.html>
4. <https://sergeygavaga.gitbooks.io/kurs-lektsii-testirovanie-programnogo-obespecheni/content/lektsiya-4-ch3.html>
5. <https://www.retail.ru/rbc/pressreleases/tsentr-povysheniya-kvalifikatsii-lider-organizatsiya-rabochego-mesta-ofisnogo-rabotnika/>
6. <https://xn--d1aux.xn--p1ai/opisanie-rabochego-mesta-programmista-na-predpriyatii/>
7. <https://studopedia.ru/8_107307_osveshchenie-pomeshcheniy-vichislitelnih-tsentrov.html>

Приложение А

Программный кодкласса Couping

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace Couping

{

public class Game : Control

{

protected int \_sizeS;

protected int \_size;

public int minSize;

public int maxSize;

protected int \_x;// расположение X

protected int \_y;// расположение Y

public int \_imSize = 150;//ширина фото высота фото

protected bool \_gameStatus = false;

public bool GameStatus

{

get

{

return \_gameStatus;

}

}

private int \_level;

public int gameLevel

{

get => \_level;

set {

if (value > 0 && value < 10)

\_level = value;

}

}

protected int X

{

get => \_x;

set { if (\_x != 0) { \_x = 0; } }

}

protected int Y

{

get => \_y;

set { if (\_y != 0) { \_y = 0; } }

}

//Метод вызывающийся при запуске игры

public void StarProcess()

{

\_gameStatus = true;

if (\_gameStatus)

{

left = gameLevel + 1;

GenerateCoup();

}

Invalidate();

}

//Метод вызывающийся при завершении игры

public void EndProcess()

{

\_gameStatus = false;

AllClear();

Invalidate();

}

public async void NextLvl()

{

AllClear();

\_gameStatus = true;

gameLevel += 1;

left = gameLevel + 1;

if (gameLevel == 4)

{

WinOrLose();

}

else

{

MessageBox.Show("Загрузка уровня");

await Task.Delay(2000);

GenerateCoup();

}

}

public void WinOrLose()

{

MessageBox.Show("You Win");

EndProcess();

}

//Метод очистки игровой зоны

public void AllClear()

{

cards1.Clear();

cards2.Clear();

cards1bg.Clear();

cards2bg.Clear();

selectedBone = -1;

selectedBone1 = -1;

}

List<int> cards1 = new List<int>();

List<int> cards2 = new List<int>();

List<int> cards1bg = new List<int>();

List<int> cards2bg = new List<int>();

//генерация наименований картинок

public void GenerateCoup()

{

bool flag = true;

bool flag2 = true;

int cc = gameLevel + 1;

while (cards1.Count != cc)

{

Random r = new Random();

int value = r.Next(0, 7);

flag = cards1.Contains(value);

if (flag == false)

{

cards1.Add(value);

cards1bg.Add(10);

}

}

while (cards2.Count != cards1.Count)

{

Random d = new Random();

int value = d.Next(0, 7);

flag = cards1.Contains(value);

flag2 = cards2.Contains(value);

if (flag)

{

if (flag2 == false)

{

cards2.Add(value);

cards2bg.Add(10);

}

}

}

}

//Метод отвечающий за прорисовку уровня

protected override void OnPaint(PaintEventArgs e)

{

int countcards = 0;

if (\_gameStatus == true)

{

GenerateCoup();

countcards = gameLevel + 1;

}

Rectangle[] rct1 = new Rectangle[countcards];

Rectangle[] rct2 = new Rectangle[countcards];

for (int j = 0; j < countcards; j++)

{

rct1[j] = new Rectangle(X + (j \* \_imSize) + j \* 20, Y, \_imSize, \_imSize);

rct2[j] = new Rectangle(X + (j \* \_imSize) + j \* 20, Y + (\_imSize) + 20, \_imSize, \_imSize);

Bitmap img1 = (Bitmap)Properties.Resources.ResourceManager.GetObject($"p{cards1[j]}");

Bitmap img2 = (Bitmap)Properties.Resources.ResourceManager.GetObject($"p{cards2[j]}");

if (j == selectedBone && flag1)

{

e.Graphics.DrawImage(img1, rct1[j]);

}

else

{

e.Graphics.DrawImage((Bitmap)Properties.Resources.ResourceManager.GetObject($"p{cards1bg[j]}"), rct1[j]);

}

if (j == selectedBone1 && flag2)

{

e.Graphics.DrawImage(img2, rct2[j]);

}

else

{

e.Graphics.DrawImage((Bitmap)Properties.Resources.ResourceManager.GetObject($"p{cards2bg[j]}"), rct2[j]);

}

}

e.Dispose();

}

public Game()

{

\_size = 20;

minSize = 200;

maxSize = 800;

\_y = 20;

\_x = 20;

}

public int iSize

{

get

{

return \_imSize;

}

set

{

if (\_imSize == 0)

{

\_imSize = 100;

}

else

{

\_imSize = value;

Invalidate();

}

}

}

bool flag1 = false;

bool flag2 = false;

int selectedBone = -1;

int selectedBone1 = -1;

int fl1;

int fl2;

public void onMouseClick(MouseEventArgs e)

{

int zi = e.X / (\_imSize + 20);

int zj = e.Y / (\_imSize + 20);

if (e.Button == MouseButtons.Left)

{

if (zi < cards1.Count && zj == 0)

{

flag1 = true;

selectedBone = zi;

fl1 = cards1.ElementAt(selectedBone);

}

if (zi < cards2.Count && zj == 1)

{

flag2 = true;

selectedBone1 = zi;

fl2 = cards2.ElementAt(selectedBone1);

}

IsCoup();

}

Invalidate();

}

public int left;

//Метод проверки одинаковых изображений

public async void IsCoup()

{

int f1 = fl1;

int f2 = fl2;

if (flag1 == flag2)

{

await Task.Delay(1000);

if (f1 == f2)

{

cards1bg[selectedBone] = 9;

cards1[selectedBone] = 9;

cards2bg[selectedBone1] = 9;

cards2[selectedBone1] = 9;

flag1 = false;

flag2 = false;

left -= 1;

if (left == 0)

{

NextLvl();

}

}

else

{

flag1 = false;

flag2 = false;

}

}

Invalidate();

}

protected override void SetBoundsCore(int x, int y, int width, int height, BoundsSpecified specified)

{

base.SetBoundsCore(x, y, width, height, specified);

Invalidate();

}

protected override CreateParams CreateParams

{

get

{

CreateParams cp = base.CreateParams;

cp.ExStyle = 0x02000000;

return cp;

}

}

}

}