МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Алтайский государственный университет»

Колледж Алтайского государственного университета

Отделение экономики и информационных технологий

ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ

(курсовая работа по дисциплине «Основы программирования»)

Выполнил студент

1 курса 283-а группы

Саморуков А.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Научный руководитель

преподаватель высшей категории

Хорохордина Е.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Работа защищена

«\_\_\_»\_\_\_\_\_\_\_\_\_2019 г.

Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2019

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc9197099)

[1 ЗНАКОМСТВО С ВИЗУАЛЬНЫМ ПРОГРАММИРОВАНИЕМ 5](#_Toc9197100)

[1.1 Определение визуального программирования 5](#_Toc9197101)

[1.2 История развития визуального программирования 8](#_Toc9197102)

[1.3 Преимущества и недостатки визуального программирования 13](#_Toc9197103)

[2 Применение визуального программирования 16](#_Toc9197104)

[2.1 Создание игр 16](#_Toc9197105)

# ВВЕДЕНИЕ

С момента создания первых программируемых машин человечество придумали более восьми тысяч языков программирования. Из года в год их число увеличивается. Некоторыми языками умеет пользоваться только небольшое число людей, которые сами их и разработали, в то время как другие языки становятся известны миллионам людей по всему свету.

На данный момент почти все программы создаются с помощью языков программирования. В теории мы можем написать программу на естественном для машины языке – бинарном коде, но это займёт много времени и будет неэффективно. Языки программирования были созданы для упрощения взаимодействия между программистом и машиной.

Каждый язык программирования, в большинстве случаев имеет больше чем одного языка-предка. Некоторые языки создаются как комбинация из элементов различных языков.

Существует множество черт, согласно которым можно классифицировать языки, при этом одних из них однозначно разделяют языки на основании технических свойств, а другие основываются на доминирующих признаках.

Данную курсовой работе я хочу посвятить одному из способов создания программ для ЭВМ – Визуальному программированию. Тема выбрана из-за того что я считаю что за визуальным программированием стоит будущее.

Тема является актуальной так как программирование всё время развивается и исходя из этого можно сделать вывод что визуальное представление информации является наиболее востребованным для изучения и применения.

Такиим образом, целью данной курсовой работы являются – определить , что такое визуальное программирование , изучить историю развития , разобрать преимущества и недостатки и изучить практическое применение визуального программирования в наши дни.

Задачи курсовой работы:

1. Рассмотреть понятие визуального программирования, изучить несколько его представителей.
2. Разобрать в каких случаях применяется визуальное программирование.

Базой для написания курсовой работы являются публицистические и аналитические материалы современных учёных.

# ЗНАКОМСТВО С ВИЗУАЛЬНЫМ ПРОГРАММИРОВАНИЕМ

## Определение визуального программирования

Визуальное программирование является в настоящее время одной из наиболее популярных парадигм программирования. Визуальное программирование состоит в автоматизированной разработке программ с использованием особой диалоговой оболочки. Рассматривая системы визуального программирования, легко увидеть, что все они базируются на объектно-ориентированном программировании и являются его логическим продолжением. Наиболее часто визуальное программирование используется для создания интерфейса программ и систем управления базами данных[1].

Начальные шаги технологии визуального программирования определяются оболочкой самой среды визуального программирования. Сначала создаются экранные формы простейшей буксировкой мыши. В инспекторе объектов производится настройка их свойств путем заполнения отдельных полей. На главную форму помимо визуальных компонент наносятся невизуальные компоненты. Формы объединяются в единый проект. Далее в соответствии со сценарием диалога программируются методы события основной и подчиненных форм. Программы "пустых" методов событий появляются в окне редактора после нажатия соответствующих клавиш или действий мыши. "Пустые" методы дополняются определенными операторами активации и дезактивации форм. По окончании начальных шагов получается работающий "скелет" программы с источниками данных из файловых баз данных и со сгенерированными формами документов, выводимых на печать. Исследователь (Browser) обеспечивает визуализацию схемы иерархии классов полученного "скелета" программы. Другими словами, технический проект реализованной части программы формируется автоматически[1].

Визуальное программирование это любой язык программирования который позволяет пользователям создавать программы с помощью перемещения графических элементов программы вместо её написания. Часто его представляют как следующий этап развития текстовых языков программирования.

Визуальное программирования основывается на идее «квадратов и стрелок», в которой квадраты или любые другие объекты на экране рассматриваются как объекты, связанные с помощью линий или дуг представляющие отношение между этими объектами.

Языки визуального программирования могут быть классифицированы в соответствии с типом использованного способа визуального выражения на языки, основывающиеся на: изображении, формах или диаграммах. Среда визуального программирования предоставляет пользователю графические элементы с которыми он может взаимодействовать по определённым правилам для построения нужной ему программы.

Главными задачами визуального программирования являются: сделать программирование доступнее для новичков и поддерживание программистов на трёх разных уровнях

* Синтаксис: Языки визуального программирования используют изображения и блоки, формы и диаграммы, для того чтобы уменьшить или избавиться в целом от возможных синтаксических ошибок помогая с построением программы. Это можно сравнить с авто исправлением в текстовых редакторах или даже с автозаменой некоторых слов.
* Семантика: Языки визуального программирования могут предоставлять некоторые механизмы которые позволят раскрыть основы программирования. Сюда входит документация встроенная в язык программирования.
* Прагматика: Языки визуального программирования поддерживают изучения того что в некоторых ситуациях означают программы. Такой уровень поддержки позволяет пользователям помещать созданные с помощью визуального программирования проекты в определённые состояния для того чтобы исследовать как на это отреагирует программа.

Текущие разработки пытаются интегрировать подход визуального программирования с языками программирования потока данных для того, чтобы иметь мгновенный доступ к состоянию программы, выходящий в отладку в режиме реального времени, или автоматическую генерацию программы и документации.

## История развития визуального программирования

Визуальное программирование – не новая концепция, она берет начало еще с 1975 года. Иногда, в силу различных причин, создать программу с помощью элементов визуального программирования намного легче, чем использовать стандартный подход с написанием исходного кода в виде текста.

Когда речь идет о визуальном программировании, первое, что приходит на ум это среда LabVIEW. Компания National Instruments, ответственная за его создание, была основа в 1976 году тремя основателями ­– Джеффри Кодоски, Джеймс Тручард и Билл Новлин в американском городе Остин, штат Техас. Основной специализацией компании являлись инструментальные средства для измерений и автоматизация производства.

В 1983 году компания National Instruments начала поиски способов сокращения времени, необходимого для программирования измерительных систем. В результате появилась концепция виртуального прибора LabVIEW – сочетания интуитивного пользовательского интерфейса лицевой панели с передовой методикой блок-диаграммного программирования, позволяющего создавать эффективные измерительные системы на основе графического программного обеспечения.

Первая версия LabVIEW увидела свет спустя десять лет после создания компании – в 1986 году (эта версия была только для Apple Mac) . Инженеры National Instruments решили пойти против обычных языков программирования и создали полностью графическую среду разработки. Основным идеологом графического подхода стал Джефф. Из года в год выпускались новые версии программы. Первой кроссплатформенной версией (включающая поддержку Windows) была третья версия, выпущенная в 1993 году. На данный момент актуальной является версия 8.6, вышедшая в прошлом году.

LabVIEW предоставляет визуальный язык программирования «G», основным назначением которого являются сбор данных, управление инструментами, индустриальная автоматизация и обработка сигналов. Это высокоспециализированный язык, плохо предназначенный для программ общего назначения, но дающий разработчику большой набор специальных инструментов.

Проекты LabVIEW относятся к области действия стандартов IEEE. А именно приложения на LabVIEW относятся к задачам измерений и автоматизации, в которых обычно ожидается короткий цикл разработки. Большинство задач состоят из сбора, анализа и представления данных. Эти особенности позволяют сформулировать требования к ТЗ, специфичные для проектов LabVIEW. По моему мнению, обязательны следующие пункты:

* Формулировка основной задачи;
* Бюджет;
* Временные рамки;
* Подробные описания параметров сбора, анализа и представления данных;
* Приоритет каждого требования;
* Методика тестирования[2].

В LabVIEW разрабатываемые программные модули называются «Virtual Instruments» (Виртуальные Инструменты) или по-простому VI. Они сохраняются в файлах с расширением \*.vi. VIs – это кирпичики, из которых состоит LabVIEW – программа. Любая LabVIEW программа содержит как минимум один VI. В терминах языка Си можно достаточно смело провести аналогию с функцией с той лишь разницей, что в LabVIEW одна функция содержится в одном файле (можно также создавать библиотеки инструментов). Само собой разумеется, один VI может быть вызван из другого VI. В принципе каждый VI состоит из двух частей — Блок-Диаграмма (Block Diagram) и Передняя Панель (Front Panel). Блок-диаграмма — это программный код (точнее визуальное графическое представление кода), а Передняя панель — это интерфейс.

Basic был создан коллективом сотрудников Дартмугского колледжа о главе с Джоном Кемени и Томасом Куртом. Это произошло в далеком 1964 году, задолго до появления персонального компьютера в его современном виде, в эпоху больших ЭВМ. BASIC расшифровывался как Beginner's All-purpose Standard Instruction Code (Все целевой стандартный код инструкций для начинающих).

Поначалу язык Basic предназначался, прежде всего, для обучения новичков принципам программирования. Visual Basic – это лучший язык программирования для новичков или любителей, позволяющий быстро начать создавать собственные приложения для Windows[3].

Дизайн «тяни и отпускай» был разработан Аланом Купером и его компанией Tripod. Microsoft заключили с Купером и его партнёрами договор по которому они должны сделать из Tripod систему программируемых форм для Windows 3.0 под кодовым именем Ruby. В нём не было языка программирования от слова совсем. Microsoft решили соединить Ruby с языком Basic для создания Visual Basic.

Visual Basic представляет собой интегрированную среду разработки, которая содержит набор инструментов, облегчающих и ускоряющих процесс разработки приложений. Причем процесс разработки заключается не в написании программы (программного кода), а в проектировании приложения. Приложение формируется средствами графического редактирования (компоновки), что позволяет свести процесс создания программного кода к минимуму.

Начиная с версии 4.0, Visual Basic поставляется в одной из трёх редакций – учебная, профессиональная и промышленная. Учебная версия позволяет создавать большинство приложений для MS Windows и Windows NT. Она включает все набиолее важные элементы управления, компоненты для работы с таблицами и базами данных. Профессиональная редакция включает в себя все возможности учебной версии, дополнительные компоненты ActiveX, средства разработки интернет-приложений, интегрированную базу данных и конструктор динамических страниц HTML.

Год за годом выпускались новые версии , но в конечном итоге поддержка Visual Basic закончилась в марте 2008 года и теперь он входит в состав Visual Studio 2017.

Ещё одним представителем визуального программирования является среда программирования Delphi. Она появилась в связи с тем, что в Windows решили что необходимо двигаться в сторону «визуализации» программирования. Эта среда была одной из первых систем RAD, первая версия была выпущена в 1995 году и была предназначена для разработки 16-разрядных приложений для Windows 3.1. Характеризуя среду программирования Delphi, о ней также говорят как о визуальной и событийно-ориентированной[4].

Среда Delphi предназначена для быстрой (RAD) разработки прикладного ПО для операционных систем Windows, Mac OS X, а также iOS и Android. Благодаря уникальной совокупности простоты языка и генерации машинного кода позволяет непосредственно, и, при желании, достаточно низкоуровнево взаимодействовать с операционной системой, а также с библиотеками, написанными на C/C++. Созданные программы независимы от стороннего ПО, как то Microsoft .NET Framework или Java Virtual Machine. Выделение и освобождение памяти управляется в основном пользовательским кодом, что, с одной стороны, ужесточает требования к качеству кода, а с другой — делает возможным создание сложных приложений с высокими требованиями к отзывчивости (работа в реальном времени). В кросс-компиляторах для мобильных платформ предусмотрен автоматический подсчёт ссылок на объекты, облегчающий задачу управления их временем жизни.

Процесс разработки в Delphi предельно упрощен. В первую очередь это относится к созданию интерфейса, на который уходит 80% времени разработки программы. Достаточно просто перетащить нужные компоненты на поверхность Windows-окна (в Delphi оно называется формой) и настраиваете их свойства с помощью специального инструмента (Object Inspector). С его помощью можно связать события этих компонентов (нажатие на кнопку, выбор мышью элемента в списке и т.д.) с кодом его обработки.

Разработчик получил в распоряжение мощные средства отладки (вплоть до пошагового выполнения команд процессора), удобную контекстную справочную систему (в том числе и по Microsoft API), средства коллективной работы над проектом.

## Преимущества и недостатки визуального программирования

С помощью визуального программирования можно описывать процессы в графическом виде, в отличии от текстового представления в обычных языках программирования, где нужно приложить дополнительные усилия, чтобы мыслить так, как это должен выполнять компьютер. Это облегчает задачу визуально представить и объяснить принцип работы вашей программы третьему лицу. Однако практически все визуальные средства нуждаются в дополнении функциями, которые нельзя представить в виде графических конструкций и требуют текстового выражения. Визуальные средства дополняются специальными программами - "скриптами", написанными на различных языках программирования.

Само по себе программирование подразумевает не только процесс написания кода, но зачастую на это тратится большая часть времени при разработке. Только представьте, сколько усилий приходится тратить на то, чтобы держать в голове множество правил и спецификаций к конкретному языку программирования, вместо того, чтобы сосредоточиться на решаемой проблеме. Поэтому гя

Часто советуют начать разработку ПО с графического описания будущей системы, ее компонентов и связей между ними, чтобы на ранних стадиях определить более выгодную структуру системы и минимизировать возможные проблемы в будущем. Графическое представление легче для понимания, чем текстовый вариант, но может иметь свои ограничения, к тому же это все равно придется переводить в понятный компилятору код. Конечно, на маленькие приложения (какими они могут казаться вначале) это не распространяется, можно сразу приступить к написанию кода, но проблема все равно остается — нужно думать в рамках определенного языка программирования. Тем более, когда вы это делаете в давно приевшемся вам императивном стиле.

Языки текстового программирования научились разделять структурные блоки кода на отдельные исходные файлы, поэтому изменение одной части системы легко слить с изменением в другом. Визуальные инструменты обычно сохраняют результат по принципу «одна диаграмма на один файл», что означает, что слияния становятся проблематичными, и еще сложнее, если семантическое значение «диффа» трудно анализировать.

Отсутствие хорошего контроля над версиями — еще один серьезный недостаток большинства инструментов визуального программирования.

Благодаря связям сам принцип построения логики с их помощью интуитивно понятен большинству людей, которые, возможно, никогда и не пробовали ничего программировать. Вот есть связь, отвечающая за перемещение, вот есть связь, которая запускает цепочку логики. Если их соединить, то объект начнёт совершать перемещение.Можно сравнить с конструктором для детей. И по такому принципу из простых действий, соединённых по порядку, получается целостный кусок логики.

Если неаккуратно проектировать программу то могут получаться так называемые «макароны» - именно так называют то безумие что получается из запутанных связей и нодов. При написании массивной логики с кучей связей велик шанс получить настоящую кашу, которую очень тяжело разобрать и в которой трудно работать. От этой проблемы практически невозможно избавиться, единственное, что остаётся – посоветовать использовать инструменты выравнивания и комментировать все блоки кода без исключения. Так шанс запутаться будет ниже

Кажется визуальное программирование должно быстрее восприниматься и легче пониматься. Но это не совсем правда. Тем, кто привык к написанию кода в классическом варианте будет непривычно, что на том же количестве пространства экрана умещается раз в 10 меньше информации. Если в обычном варианте вы просто читаете строчку за строчкой, то в визуальной среде вам понадобится постоянно перемещать рабочее пространство, чтобы изучить даже небольшое количество кода.

Все вышеописанные недочёты компенсируются удобством и скоростью создания кода или уже придуманы способы их устранения. Так что каждый пользователь решает сам насколько удобна эта система.

# Применение визуального программирования

## Создание игр

Визуальное программирование также используется для создания игр. Примером может послужить Unreal Engine 4. Это игровой движок, разработанный и до сих пор поддерживаемый компанией Epic Game. Движок написан на языке C++ и позволяет создавать игры для большинства операционных систем и платформ.

Blueprints это система визуального скриптинга в Unreal Engine, основанная на концепте использования узлов интерфейса для создания элементов геймплея внутри Unreal Editor. Как и многие скриптовые языки, он используется для определения объектно-ориентированных классов или объектов в движке.При использовании UE4, вы часто будете замечать что объекты, определённые с при помощи Blueprint, называются просто «Blueprints».

Эта система чрезвычайно гибкая и мощная, поскольку она даёт дизайнерам возможность использовать практически весь спектр концепций и инструментов, которые обычно доступны только программистам. В дополнении к этому доступна специфичная разметка Blueprint позволяющая программистам создавать базовые системы, которые в будущем могут быть продолжены дизайнерами.

Система предоставляет визуальный подход к скриптовому языку. Таким образом она разделяет многие нюансы обычных текстовых скриптовых языков, такие как типизированные переменные данных, массивы, структуры и т.д. Также поток выполнения работае как и должен в типичном скриптовом языке, хотя Blueprints требуют явного линейного выполнения для каждого узла.

По своей сути Blueprints это визуально скриптовые дополнения для создаваемой вами игры. С помощью соединения узлов, событий, функций и переменных линиями возможно создать сложные игровые элементы.

Blueprints работает используя графики узлов для различных целей – строительства объектов, отдельных функций и общих событий игрового процесса, которые являются особенными для каждого Blueprint для внедрения поведения и остального функционала.

Сейчас, используя Blueprint, можно почти полностью сделать игру и не писать ни одной строчки кода C++. Считается что главная его сила в скорости итерации. С того момента, как вы что-то изменили в программе и нажали кнопку «play», проходит полсекунды-секунда и вот вы уже проверяете, что изменилось. Благодаря отличной интеграции с движком скорость разработки повышается.

По моему мнению, создание игр — это та самая сфера деятельности, где визуальное программирование можно использовать по-настоящему эффективно. Если взять простую головоломку или аркаду, её можно разбить на простейшие геймплейные блоки, которые разработчик переписывает снова и снова в каждом проекте (ну, или просто копирует). Так почему бы не создать библиотеку этих простейших блоков? Ведь именно с ними вы и работаете в Blueprint. Более того, уже в данный момент можно разделить функционал между программистами и геймдизайнерами. Первые будут писать «внутренности» для блоков, а вторые - соединять их в готовый функционал.