МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

РАЗРАБОТКА ЧАСТЕЙ ИГРЫ «АРКАНОЙД» С ПРИМЕНЕНИЕМ ТЕХНОЛОГИИ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ MPI

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Шлюпкина Павла Владимировича

Научный руководитель
доцент, к.ф.-м.н. _____ К.П. Савина

подпись, дата

Зав. кафедрой:
к.ф.-м.н., доцент _____ М.В. Огнева

подпись, дата

Саратов 2020

ВВЕДЕНИЕ

Актуальность игры не уменьшилась, как для пользователей, желающих развить реакцию, так и для обучения разработке программного обеспечения. Arkanoid реализуема на многих современных языках программирования. Подходы к изучению и цели обучения могут быть разные. Во-первых, данную игру часто разрабатывают для лучшего понимания синтаксиса языка программирования, умению работать с математическими вычислениями и с пользовательским интерфейсом. Во-вторых, «Арканойд» реализуют для изучения элементов параллелизма, например, при обучении программированию на языке Java данную игру стали рассматривать, как задание для изучения параллелизма с технологией многопоточности. Данный подход в приоритете, потому что в игре участвуют отдельные элементы, а именно: блоки, шары, платформа, которые работают независимо и параллельно. Пока один объект движется или статичен, другие элементы также продолжают работать.

Целью данной бакалаврской работы является разработка параллельной версии игры «Арканойд», с использованием технологии многопоточности и технологии MPI обмена сообщениями между отдельными потоками. Для достижения данной цели должны быть решены следующие задачи:

1. изучить технологию создания параллельных программ на основе многопоточности и возможности, предоставляемые языком программирования C#, для реализации многопоточности;
2. изучить технологию создания параллельных программ посредством MPI и возможности, предоставляемые языком программирования C#, для реализации MPI;
3. разработать управляющую и информационную части игры «Арканойд» с выделением потоков и выводом о них информации используя технологии многопоточности и MPI.

Выбор данной темы состоит в том, что параллельные технологии привлекают к себе внимание разработчиков игр и программного обеспечения, чтобы ускорить работу своих продуктов.

Методологические основы разработки образовательной игры виртуальной реальности представлены в работах таких авторов как: Frank Nielsen [1], Roman Trobec [2], Pitt-Francis J. [3], Albahari J., Albahari B. [4], Крюков В.А. [5], Богачев К.Ю. [6], Gregor D. [7], Симон Р. [8], Башашин М. В., Сапожникова Т. Ф. [9].

Практическая значимость бакалаврской работы. Реализацию игры «Арканойд», выполненную в данной работе, можно рассматривать, как обучающее задание для студентов, используемое как в курсах программирования для платформы .NET с целью изучения работы параллельных алгоритмов и их применения в разработке игр, так и для демонстрации независимой работы потоков в курсах параллельного программирования.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и пяти приложений. Общий объем работы – 74 страниц, из них 51 страница – основное содержание, включая 13 рисунков и 1 таблицу, список использованных источников информации – 24 наименования.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Технологии параллельного программирования» посвящен изложению теоретического материала о технологиях параллельного программирования. Предоставлена информация о потоках, процессах и многопоточности. Приведены примеры использования технологий System.Threading и MPI.NET.

Процесс можно рассматривать как способ объединения родственных ресурсов в одну группу. У процесса есть адресное пространство, содержащее текст программы, и данные, а также другие ресурсы.

Поток выполнения, от английского thread, нить, – наименьшая единица обработки, исполнение которой может быть назначено ядром операционной системы. В большинстве случаев поток выполнения находится внутри процесса. Несколько потоков выполнения могут существовать в рамках одного и того же процесса и совместно использовать ресурсы, например, память.

Процесс состоит хотя бы из одного потока: в операционной системе каждому процессу соответствует адресное пространство и обязательно один поток выполнения, но сам процесс ничего не исполняет, он просто служит контейнером потоков.

Многопоточность – это свойство платформы или приложения, состоящее в том, что процесс, порожденный в операционной системе, может состоять из нескольких потоков, выполняющихся параллельно. При выполнении некоторых задач такое разделение позволит эффективно использовать ресурсы компьютера. Многопоточность означает, что внутри процесса могут быть несколько потоков [1,2,3].

К достоинствам многопоточной реализации той или иной системы перед многозадачной можно отнести следующее:

- упрощение программы в некоторых случаях за счет использования общего адресного пространства.

- меньшие относительно процесса временны́е затраты на создание потока.

К достоинствам многопоточной реализации той или иной системы перед однопоточной можно отнести следующее:

- упрощение программы в некоторых случаях, за счёт вынесения механизмов чередования выполнения различных слабо взаимосвязанных подзадач, требующих одновременного выполнения, в отдельную подсистему многопоточности.
- повышение производительности процесса за счет распараллеливания процессорных вычислений и операций ввода-вывода.

На многопроцессорных компьютерах многопоточность реализована как смесь квантования времени и подлинного параллелизма, когда разные потоки выполняют код на разных вычислительных элементах. Необходимость квантования времени все равно остается, так как операционная система должна обслуживать как свои собственные потоки, так и потоки других приложений.

Поток вытесняется, когда его выполнение приостанавливается из-за внешних факторов типа квантования времени. В большинстве случаев поток не может контролировать, когда и где он будет вытеснен.

В приложениях Windows Forms, когда главный поток занят длительными вычислениями, он не может обрабатывать сообщения от клавиатуры и мыши, и тогда приложение перестает откликаться, поэтому следует запускать отнимающие много времени задачи параллельно с основным потоком. В этом случае модальная диалоговая форма продолжает получать сообщения, пока задача выполняется в фоновом потоке.

В случае приложений без пользовательского интерфейса, например, служб Windows, многопоточность имеет смысл, если выполняемая задача может занять много времени, так как требуется ожидание ответа от другого компьютера (сервера приложений, сервера баз данных или клиента). Запуск

такой задачи в отдельном рабочем потоке означает, что главный поток немедленно освобождается для других задач.

Клиентское приложение C# запускается в одном потоке, который создается автоматически библиотекой CLR (Common Language Runtime) и операционной системой (главный поток «Main»). Многопоточность обеспечивается путем создания дополнительных потоков.

Язык C# имеет встроенную поддержку многопоточности, а среда .NET Framework предоставляет классы для работы с потоками, что в совокупности помогает гибко и правильно реализовывать многопоточность в проектах [4,5].

Основной функционал для использования потоков в приложении сосредоточен в пространстве имен System.Threading. В нем определен класс, представляющий отдельный поток – класс Thread, инкапсулирующий в себе работу с потоком и определяющий ряд методов и свойств, которые позволяют управлять потоком и получать информацию о нем. При создании экземпляра этого класса выделяется память под новый поток, в котором будут происходить необходимые действия.

В модели передачи сообщений параллельная программа выполняется на множестве процессов, каждый из которых имеет свое собственное адресное пространство. Обмен данными и синхронизация между процессами производится посредством передачи сообщений. MPI программы – это множество параллельных взаимодействующих процессов, у каждого процесса есть свое адресное пространство, и никаких общих переменных или общих данных в MPI нет. Процессы выполняются параллельно и взаимодействуют между собой посредством приема и передачи сообщений.

После инициализации MPI все активные процессы объединяются в общую группу с единым коммуникатором Communicator.world. Для работы с группами процессов используем Intracommunicator, описывающий область для связи некоторой группы процессов.

Существуют два основных свойства коммуникаторов, используемые каждой программой MPI: ранг процесса в коммуникаторе, который идентифицирует этот процесс, и размер коммуникатора, который обеспечивает количество процессов в коммуникаторе [6,7].

В MPI.NET операции «точка-точка» являются основными операциями обмена, позволяющими отправлять сообщения от одного процесса к другому в пределах коммуникатора. Каждое сообщение имеет процесс-источник и процесс-приемник. Существует два вида операций обмена: блокирующего и неблокирующего.

Написание параллельных программ можно полностью реализовать через парный обмен, однако операции коллективного взаимодействия имеют по отношению к ним ряд преимуществ. Реализации MPI обычно содержат оптимизированные алгоритмы для коллективных операций, в которых используются знания топологии сети и аппаратного обеспечения.

В операциях коллективного взаимодействия процессов участвуют все потоки коммуникатора. Соответствующая процедура должна быть вызвана каждым потоком, возможно, со своим набором параметров. Возврат из процедуры коллективного взаимодействия может произойти в тот момент, когда участие потока в данной операции уже закончено. Как и для блокирующих процедур, возврат означает, что разрешен свободный доступ к буферу приема или отправки, но не означает ни того, что операция завершена другими потоками, ни того, что она ими начата (если это возможно по смыслу операции) [8,9].

Во **втором разделе «Разработка структуры классов и реализация программных модулей игры «Арканойд»** описана структура классов реализованных модулей игры. Присутствует постановка задачи и описание аспектов игры. Предоставлена реализация программных модулей игры.

Разработка и реализация частей игры «Арканойд» выполнялась с применением параллелизма, поддерживаемого в языке программирования C#, а также, предоставляемого стандартом MPI.

Приложение игры в виде фрейма, должно содержать игровую область, представляющую из себя ограниченный прямоугольник, управляющую и информационную части. В игровой области присутствует «дощечка», в виде трапеции, ожидающая отражения шариков, двигающаяся вдоль нижней границы области, и шары, отражающиеся от границ области и «дощечки».

Каждый шар имеет свою скорость, выбранную случайным образом из заданного в управляющей части диапазона. При старте все шары разлетаются в случайном направлении из центра игровой области и со случайной скоростью. Если пять последних отскоков шарика двигается по одной и той же траектории, то он случайным образом меняет свое направление. Управление перемещением дощечки должно осуществляться с помощью её захвата мышью. «Дощечка» движется только при удерживаемой клавише мыши – то есть пользователь должен попасть курсором мыши в дощечку, «захватить» дощечку нажатием клавиши мыши и после этого передвигать её. Как только кнопка мыши отпущена – «дощечка» освобождается и перемещать её можно будет только после очередного «захвата».

В управляющей части игры присутствуют поля с заданием диапазона скоростей в некоторых условных единицах, определенных в реализации, текстовое поле для задания количества шариков, кнопки «Старт», «Стоп», «Пауза». При нажатии клавиши «Пауза» игра останавливается с возможностью продолжить в будущем.

В информационной части игры выводится время и очки игрока. Каждая попытка записывается в результирующую таблицу. Очки вычисляются, исходя из текущей скорости к максимально возможной: отношение текущего количества шаров к начальному, +1% за каждый отскок от трапеции. Очки и время выводится в течение игры каждую секунду.

В игровой области шарик, долетевший до нижней границы области считается неотраженным (упущенным) и должен исчезать. Игра считается законченной, когда все шары упущены. Шары являются прозрачными друг для друга (т.е. шары не отражаются друг от друга). Шары отражаются от

границ и от «дощечки» (в том числе и от её боковых сторон). С течением времени увеличивается скорость движения шариков. Для каждого шара вводится своя полярная система координат и при отскоке соответственно изменяется угол его движения. Угол движения будет меняться по формуле Френеля (рисунок 3), а именно «угол отражения равен углу падения». Нужно учитывать, что при отскоке и изменении градусов, координат шаров используется библиотека Math для выполнения математических операций, в которой отсчёт координат идёт, как в обычной системе координат, от которой отличается система координат, используемая для отображения движения шариков на игровом поле.

Точка пересечения движения шарика и доски высчитывается по формуле $y = k * x + b$, где k – угол наклона шарика по направлению к платформе, а b – расстояние до точки пересечения с осью Y.

В классе Program находится точка входа приложения. При создании Windows Form был сгенерирован код, также указываем среду MPI, благодаря которой будет работать приложение. Запускаем форму игры Frm.

Класс Frm работает со всеми классами, ведь в игровом поле задействованы все элементы игры, а именно блоки, трапеция, шары. Для стабильной и корректной работы каждый шар будет потоком. Количество потоков зависит от количества шаров, выбранных пользователем. Поэтому в данном классе будет подключена многопоточность. Трапеция будет работать с процессами, поэтому в классе Trapeze будет подключена библиотека MPI.

В управляющей части игры пользователь выбирает количество шаров и начальную скорость. Каждый шар реализован в виде отдельного потока, осуществляющего свою прорисовку. Так же реализованы методы, которые следят за временем игры и за очками, набранными игроком. Все они начинают свою работу при старте игры и завершаются при ее окончании. Также присутствуют кнопки начала и остановки игры. Информационная часть выводит результаты предыдущего запуска игры пользователем (время и очки).

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены технологии создания параллельных программ, предоставляемые языком программирования C#, на основе многопоточности System.Threading и посредством MPI.NET.

С использованием изученных технологий параллельного программирования на основе пространства имен System.Threading в C# и библиотеки MPI.NET, реализована игра «Арканойд» с применением Windows Forms, в которой игрок управляет дощечкой, отбивает шарики и шарики сбивают блоки с целью уничтожить все доступные блоки. Для игровых объектов в виде шариков выделяются отдельные потоки, которые выполняют прорисовку элементов на экране. Все потоки для шариков создаются из основного потока при старте игры и завершаются при ее окончании. Главный поток следит за временем игры и за очками, набранными игроком. На данный момент в общем доступе нет реализации игры «Арканойд» на языке C# с применением технологий параллелизма System.Threading или MPI.NET. Библиотека MPI для языка C# предоставлена Индианским университетом с 2008 года, крайнее обновление библиотеки проводилось в 2018 году. Поэтому данную реализацию, представленную в работе, можно рассматривать, как обучающее задание для разработчиков программного обеспечения на платформе .NET с целью изучения работы параллельных алгоритмов и их использования в разработке игр.

Основные источники информации:

1. Frank Nielsen. Introduction to HPC with MPI for Data Science / Frank Nielsen: Undergraduate Topics in Computer Science, 2016.

2. Roman Trobec. Introduction to Parallel Computing / Roman Trobec, Boštjan Slivnik, Patricio Bulić, Borut Robič: Undergraduate Topics in Computer Science, 2018.

3. Pitt-Francis J. Guide to Scientific Computing in C++ / Joe Pitt-Francis, Jonathan Whiteley, 2012.

4. Albahari J., Albahari B. C# 3.0 in a Nutshell, 3rd Edition / Gravenstein Highway North, Sebastopol: O'Reilly Media. 2007.

5. Крюков В.А. Разработка параллельных программ для вычислительных кластеров и сетей: Информационные технологии и вычислительные системы, 2003.

6. Богачев К.Ю. Основы параллельного программирования/ Богачев К.Ю. Москва: Бином, 2003.

7. Gregor D. MPI.NET Tutorial / D. Gregor. — Indiana University: Open Systems Laboratory, 2008.

8. Симон Р. C# для профессионалов. / Симон Р., Олли К. Москва: Лори, 2003.

9. Башашин М. В., Сапожникова Т. Ф., Работа с группами и коммуникаторами в MPI. / Башашин М. В., Сапожникова Т. Ф., Земляная Е.В.: Группа гетерогенных вычислений HybriLIT. 2017.