Частное общеобразовательное учреждение

«Православная гимназия Кирилла и Мефодия»

Проект

по информатике

**ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В ПОВЕДЕНИИ ПЕРСОНАЖЕЙ КОМПЬЮТЕРНЫХ ИГР**

Выполнил

ученик 10 класса

Черемушкин Роман Евгеньевич

Руководитель работы:

Методист

Чёрная Елена Сергеевна

2025 год

**ОГЛАВЛЕНИЕ**

[**ГЛАВА 1. ПОНЯТИЕ И ИСТОРИЯ ИЗУЧЕНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА, МАТЕМАТИЧЕСКАЯ МОДЕЛЬ** 5](#_Toc188788405)

[**1.1. Определение интеллекта. Стратегии поведения живых существ** 5](#_Toc188788406)

[**1.2. Теоретическое математическое обоснование данной работы** 6](#_Toc188788407)

[**1.2.1. Среда для экспериментов** 7](#_Toc188788408)

[**1.2.2. Математическое описание стратегий поведения** 8](#_Toc188788409)

[**1.3. Выводы** 10](#_Toc188788410)

[**ГЛАВА 2. РАЗРАБОТКА ПРОГРАММЫ. АНАЛИЗ ЭФФЕКТИВНОСТИ СТРАТЕГИЙ** 11](#_Toc188788411)

[**2.1 Разработка среды** для взаимодействия персонажей 11](#_Toc188788412)

[**1.4. Разработка стратегий** 12](#_Toc188788413)

[Выводы по второй главе 16](#_Toc188788414)

[ЗАКЛЮЧЕНИЕ 16](#_Toc188788415)

[СПИСОК ЛИТЕРАТУРЫ 17](#_Toc188788416)

[ПРИЛОЖЕНИЯ 18](#_Toc188788417)

[Приложение 1. Видео 1 с визуализацией стратегий. 18](#_Toc188788418)

[Приложение 2. Видео 2 с визуализацией стратегий. 18](#_Toc188788419)

[Приложение 3. Ссылка на исходный код 18](#_Toc188788420)

**ВВЕДЕНИЕ**

В последние десятилетия искусственный интеллект получил широкое распространение благодаря развитию компьютерных технологий. Компьютеры интегрируют в различные устройства: бытовую технику, транспорт от автомобилей до самолетов и кораблей, беспилотные летательные аппараты. От многих устройств требуется интеллектуальное поведение. Например, летательный аппарат можно научить автоматически избегать препятствий.

Применение искусственного интеллекта напрямую связано и с развитием робототехники. Первым шагом в создании робота является создание компьютерной модели его поведения.

Также компьютерные модели поведения персонажей широко используют в компьютерных играх, поскольку это популярный досуг многих людей. И так как в играх используется много подходов к созданию моделей поведения персонажей, управляемых искусственным интеллектом, и они протестированы большим количеством пользователей, то эти подходы зачастую применяют и для реальных роботов и устройств. Поэтому актуальность нашей работы распространяется и на игровую индустрию, и на применение управляемых искусственным интеллектом моделей поведения роботов и приборов в быту и промышленности.

**Цель работы** ― изучить процесс создания искусственного интеллекта. Для этого планируется создать компьютерную модель взаимодействия двух персонажей.

Для достижения цели необходимо решить ряд задач:

Создать среду, в которой будет взаимодействовать герой и враг.

Разработать несколько стратегий героя и врага.

Проанализировать работу этих стратегий в нашей среде.

**Объект исследования** ― искусственный интеллект персонажей компьютерной игры, управляемых с помощью алгоритмов. Один из них, герой - убегает, а второй, враг - догоняет. Задача героя продержаться на поле 15 секунд. Если за 15 секунд враг коснулся героя - победил враг. Если же, герой продержался, победил герой. Действие происходит на ограниченном прямоугольном 2D поле.

**Предметом исследования** являются автоматически заданные стратегии поведения героя и врага. Эти стратегии по сути - функции выбора скорости и угла движения объекта в заданных временных точках.

Персонажи данной работы, “Герой” и “Враг”, обладают искусственным интеллектом, который задан в виде функции, вычисляющей направление движения и скорость персонажа в определенные моменты времени. Подбор параметров для функции для них позволит исследовать принципы работы искусственного интеллекта.

**Методы исследования:**

― компьютерное моделирование с использованием среды разработки Godot и языка программирования C#,

― анализ и сравнение эффективностей стратегий персонажей, заданных искусственным интеллектом.

**В результате исследования** создана модель взаимодействия героя и врага, разработаны и проанализированы несколько стратегий их поведения.

**Практическая значимость** данной работы заключается в том, что ученик, ознакомившийся с этой работой, может применить свои навыки для программирования поведения реальных устройств.

# **ГЛАВА 1. ПОНЯТИЕ И ИСТОРИЯ ИЗУЧЕНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА, МАТЕМАТИЧЕСКАЯ МОДЕЛЬ**

## **Определение интеллекта. Стратегии поведения живых существ**

В толковом словаре Ожегова его определяют, как “ум, мыслительная способность, умственное начало у человека.”

В большом психологическом словаре дают более подробное определение: 1) общая способность к познанию и решению проблем, определяющая успешность любой деятельности и лежащая в основе др. способность;

2) система всех познавательных (когнитивных) способностей индивида: ощущения, восприятия, памяти, представления, мышления, воображения;

3) способность к решению проблем без проб и ошибок «в уме». Понятие И. как общая умственная способность применяется в качестве обобщения поведенческих характеристик, связанных с успешной адаптацией к новым жизненным задачам.

Таким образом, интеллект живых существ -это способность к мышлению, позволяющая избежать ошибок, лучше приспособиться к среде, действовать более эффективно.

Так, простейшие организмы, не обладающие интеллектом, например, амёба, имеют ограниченное количество стратегий взаимодействия с внешней средой и жить могут только в определенных условиях (воде определенной температуры).

Организмы, приобретшие на пути эволюции мышление, имеют гораздо более сложное поведение, которое позволило им приспособиться жить в разнообразных и изменяющихся условиях. Для этого у них сформировалась нервная система. Самое сложное поведение и самый развитый интеллект ― у человека. Поэтому он может жить в любом климате и оказывать наибольшее влияние на окружающую среду.

Методы машинного обучения позволяют одним автоматам распознавать устную речь и записывать ее подобно секретарям-машинисткам прошлых лет, а другим – точно идентифицировать лица или отпечатки пальцев среди десятков миллионов других и обрабатывать тексты, написанные на естественных языках. Благодаря этим же методам самостоятельно движутся автомобили, компьютеры лучше врачей-дерматологов диагностируют меланомы по фотографиям родинок, сделанных с помощью сотовых телефонов, роботы воюют вместо людей; а конвейеры на заводах все больше автоматизируются.

Самообучающиеся интеллектуальные системы широко применяются практически во всех сферах, особенно в промышленности, банковском деле, страховании, здравоохранении и обороне. Многие рутинные процессы теперь можно будет автоматизировать, что преобразит наши профессии и, в конечном итоге, устранит некоторые из них.

## **Теоретическое математическое обоснование данной работы**

В данной работе мы создаем двух персонажей, “Герой” и “Враг”, обладающих искусственным интеллектом. Их поведение задается в виде функции, вычисляющей направление движения и скорость персонажа в определенные моменты времени. Подбор параметров для функции для них позволит исследовать принципы работы искусственного интеллекта.

Для проведения экспериментов нам необходима платформа. Она состоит из математической модели эксперимента а также из реализации, описанной во второй части данной проектной работы.

## **Среда для экспериментов**

Зададим прямоугольное поле, каждая точка в котором имеет координаты x, y.

Враг представляет собой объект с координатами *xв, yв*, Герой представляет собой объект с координатами *xг, yг.* Герой и Враг также имеют направление и скорость: Направление героя - *αг, αг є [0, 2π).* Скорость героя дискретна *vг, vг=1,2,3,...,20.* Направление врага - *αв, αв є [0, 2π).* Скорость врага - *vв, vв=1,2,3,...,20.*

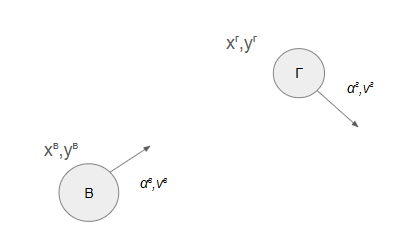


Рис. 1. На данном рисунке изображены объекты Герой(Г) и Враг(В). Враг расположен в точке с координатами *xв, yв* а герой в точке *xг, yг.* Направление движения героя - *αг.* Скорость героя *vг.* Направление движения врага *- αв.* Скорость врага - *vв.*

Герой и враг не могут выходить за пределы поля. Столкновение героя и врага (победа врага) определяется как: *d(г,в)<dmin. d(г,в)2=dx2+dy2. dx=xг-xв, dy=yг-yв.*

Временная шкала дискретна. *t=t0…tN. ti=i\*dt. dt=100* миллисекундам. У объекта (врага или героя) есть ограниченное количество энергии S. Данная энергия восполняется при отсутствии движения и тратится когда объект движется. *Sвi+1=Sвi+5-vвi.* Герою бежать легче, поэтому *Sгi+1=Siг+6-vгi*.Здоровье не может быть больше 100 и меньше 0. Если после вычисления здоровье оказалось больше 100, оно округляется до 100. Если после вычисления здоровье меньше 0, то скорость должна быть выбрана так чтобы здоровье было равно 0.

*if (Si+1>100) => Si+1=100*

*if (Si+1<0) => vi =Si+5*

Скорость и угол (героя и врага) можно изменить только в момент времени *ti* . В промежутках между *(ti,ti+1)* менять скорость и угол нельзя. Таким образом, положение героя в момент времени *ti+1* задается по формулам

*xгi+1=xгi+vгi\*cos(αгi). yгi+1=yгi+vгi\*sin(αгi).*

Положение врага в момент времени *ti+1* задается по формулам

*xвi+1=xвi+vвi\*cos(αвi). yвi+1=yвi+vвi\*sin(αвi).*

В момент времени ti нам необходимо программным способом задать значение скорости и угла. То есть нам необходимо задать функцию *(α,v)=f(xг,yг,xв,yв,S)*, Где S - здоровье героя или врага в зависимости от того кто вызывает функцию.

Функции для героя и врага различны. Эти функции авторы будут выбирать и проводить с ними эксперименты.

## **Математическое описание стратегий поведения**

Далее разработаем несколько стратегий. Приведем здесь некоторые примеры простых стратегий.

Стратегия героя 1 - герой движется по кругу радиуса R с постоянной скоростью v1=3. При приближении врага на расстояние меньше D герой ускоряется до *v2=10.*

Стратегия врага 1 - враг движется в направлении героя со скоростью *v1=4.* При приближении к герою на расстояние D2 враг делает бросок со скоростью 0*v2=15.*

Стратегия врага 2 - пусть *xв, yв* - координаты врага, а *xг, yг* - координаты героя. Враг может предположить в какой точке будет находиться герой в следующий момент времени следующим образом: сохраним координаты героя из предыдущего момента времени в переменные xгп, yгп. Таким образом, на каждом “шаге” (моменте времени) мы имеем 2 пары координат: *xг*, *yг*; xгп, yгп. Затем вычислим предполагаемые координаты героя через несколько шагов:

*yбг* = *yг* + (*yг* - *yгп*)

*xбг* = *xг* + (*xг* - *xгп*)

*yбг2* = *yг* + 2 \* (*yг* - *yгп*)

*xбг2* = *xг* + 2 \* (*xг* - *xгп*)

Теперь вычислим угол наклона, который необходим для результата нашей стратегии

*d1* = arctg *yг*/*xг*

*d2* = arctg *yбг*/*xбг*

*d3* = arctg *yбг2*/*xбг2*

В результате нашей стратегии будут выданы определенные угол и скорость в зависимости от расстояния до героя: если dx < d1 или dy < d1, то вернуть *d* = *d1*, v = 15; если dx < d2 или dy < d2, то вернуть *d* = *d2*, v = 10; иначе вернуть *d* = *d3*, v = 5. В этих формулах dx = |xв - xг|; dy = |yв - yг|, а d1 и d2 - это константы.

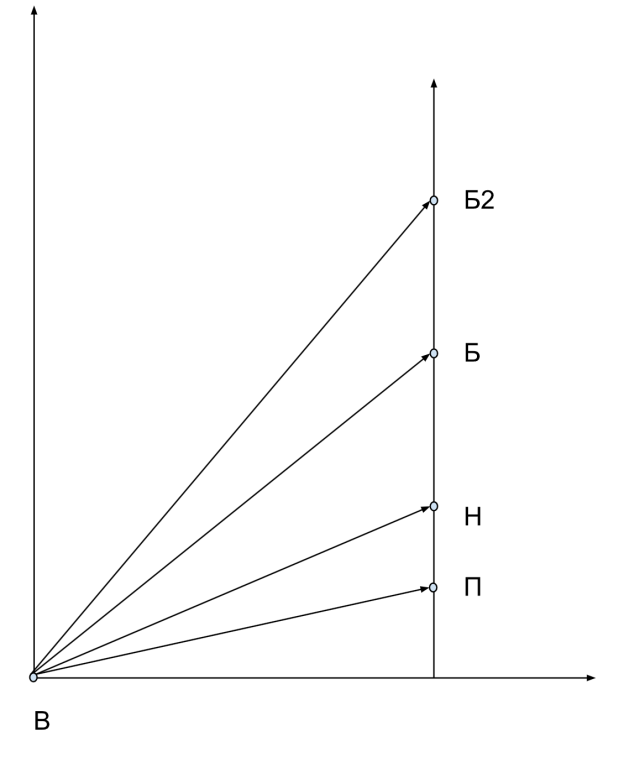


Рис 2. На данном рисунке изображена позиция врага (В) и позиции героя: П - предыдущие координаты героя. Н - настоящие координаты героя. Б и Б2 предсказанные координаты героя, к которым будет двигаться враг. К какой конкретно координате двигаться Б или Б2 будет определяться в зависимости от расстояния до героя.

## **Выводы**

Таким образом, мы выявили, что интеллект позволяет эффективнее приспосабливаться к окружающей среде. Например, поэтому люди в течение многих лет используют вычислительные мощности компьютеров и их способность работать с данными для решения различных задач от бытовых до освоения космоса.

Мы описали математическую модель игры, в которой персонажи “герой” и “враг” обладают искусственным интеллектом и стратегии взаимодействия между ними.

# **ГЛАВА 2. РАЗРАБОТКА ПРОГРАММЫ. АНАЛИЗ ЭФФЕКТИВНОСТИ СТРАТЕГИЙ**

Исходный код программы, написанной в ходе этой работы представлен в [Приложении 3](#_6c7piaqe8url).

## **2.1 Разработка среды** для взаимодействия персонажей

В качестве платформы, на которой мы реализовали стратегии, использована среда разработки Godot.

В Godot присутствуют объекты типа CharacterBody2D. Этот тип объектов содержит небольшую физическую модель: Можно задать объекту скорость и направление. При этом объект будет двигаться в этом направлении.

**public partial class Game : Node**

**{**

**public override void \_Ready()**

**{**

**Rotate(Mathf.Pi/2);**

**Velocity = Transform.X \* 5;**

**}**

**public override void \_PhysicsProcess(double delta)**

**{**

**MoveAndSlide();**

**}**

**}**

Рис. 3. (Листинг 1). На данном листинге приведен пример программы на Godot. Этот скрипт позволяет персонажу двигаться с фиксированной скоростью в направлении *Pi/2*. Функция \_Ready вызывается один раз при запуске программы. Функция \_PhysicsProcess или аналогичная ей функция \_Process вызывается с высокой частотой (около 60 раз в секунду. Внутри функции \_PhysicsProcess вызывается функция MoveAndSlide, которая встроена в CharacterBody2d и выполняет физическое движение.

В начале работы автор создал проект в среде GODOT. В нем создал два персонажа типа CharacterBody2D. Далее создал стратегии персонажей: чтобы один двигался навстречу другому а другой крутился на месте. **TODO** написать по подробнее и научно.

Далее, была спроектирована структура классов описанная в приложении 4. **TODO**

## **Разработка стратегий**

У нас есть несколько стратегий - для врага и для героя.

Стратегии врага:

1. Движение с постоянной скоростью в направлении героя
2. Стратегия забегания вперед.

Стратегия героя:

1. Движение по кругу.

**Стратегия врага 1. Движение с постоянной скоростью в направлении героя.**

Вычислим направление на героя. Пусть float x, float y - координаты врага, а float otherX, float otherY - координаты героя. Угол вычисляется по формуле:

float dx = otherX - x;

float dy = otherY - y;

float alpha = Mathf.Atan2(dy, dx);

Функция Mathf.Atan2 вычисляет угол наклона вектора (dx, dy) в радианах. Соответственно, как результат стратегии, возвращается пара значений (alpha, S) где S = 5 - константа.

**Стратегия врага 2. Стратегия предугадывания движения героя.**

Пусть float x, float y - координаты врага, а float otherX, float otherY - координаты героя. Враг может предположить, в какой точке будет находиться герой в следующий момент времени: сохраним координаты героя из предыдущего момента времени в переменные float lastotherX, float lastotherY. Таким образом, на каждом шаге мы имеем 2 пары координат: otherX,otherY; lastotherX, lastotherY. Затем вычислим предполагаемые координаты героя через несколько шагов (Mdx, Mdy; Ldx, Ldy):

float Mdx = otherX - x + 4\*(otherX - lastotherX);

float Mdy = otherY - y + 4\*(otherY - lastotherY);

float Ldx = otherX - x + 8\*(otherX - lastotherX);

float Ldy = otherY - y + 8\*(otherY - lastotherY);

Теперь вычислим угол наклона, который необходим для результата нашей стратегии

float Shortalpha = Mathf.Atan2(otherY - y, otherX - x);

float Longalpha = Mathf.Atan2(Ldy, Ldx);

float Midlalpha = Mathf.Atan2(Mdy, Mdx);

И расстояние до героя:

float distanceX = Mathf.Abs(otherX - x);

float distanceY = Mathf.Abs(otherY - y);

В результате нашей стратегии будут выданы определенные угол и скорость в зависимости от расстояния до героя:

if (distanceX > 220 | distanceY > 220)

{

alpha = Longalpha;

speed = 5;

}

else if (distanceX > 145| distanceY > 145)

{

alpha = Midlalpha;

speed = 10;

}

else

{

alpha = Shortalpha;

speed = 15;

}

**Стратегия героя 1. Движение по кругу.**

Герой движется по кругу фиксированного радиуса с скоростью S обратно пропорциональной расстоянию до врага.

В функции инициализации контроллера вычислим координаты центра

cx = x0 + (xMax - x0) / 2;

cy = y0 + (yMax - y0) / 2;

где x0, y0 координаты нижнего левого угла, а xMax, yMax координаты верхнего правого угла поля.

Вычислим направление, перпендикулярное направлению на центр.

float dx = cx - x;

float dy = cy - y;

float alpha = Mathf.Atan2(dy, dx) - Mathf.Pi / 2.0f + B \* v;

В этом направлении учтена поправка приближающая героя к центру (B\*v, где B=0.005f, а v - скорость). Поправка нужна, так как герой проходит по этому перпендикуляру некоторое расстояние (см Рис 5). Скорость героя v задается следующим образом.

float dxv = otherX - x;

float dyv = otherY - y;

float dvrag = Mathf.Sqrt(dxv \* dxv + dyv \* dyv);

if (dvrag < 20) dvrag = 20; // ограничение позволяющее избежать деления на 0.

float v = A / dvrag;

if (v > 20) v = 20;

if (health < v) v = 0;

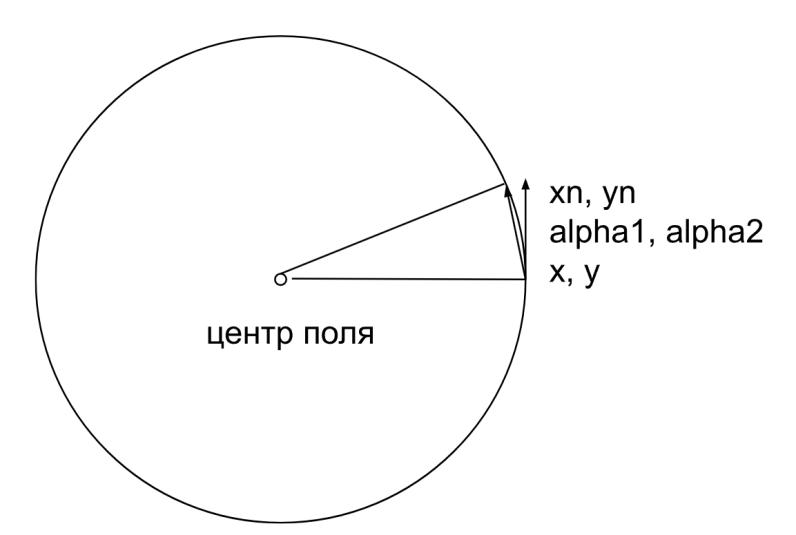


Рис. 6. На данном рисунке изображены два направления. alpha1 направление героя перпендикулярное направлению на центр поля. alpha2 направление с поправкой, которое позволяет герою двигаться по кругу. Направление alpha2 используется в качестве результата стратегии движения по кругу.

**Анализ результатов**

Мы разработали запуск стратегий:

1. Герой - стратегия движения по кругу.
2. Враг - 2 стратегии:  
   ― стратегия движения с постоянной скоростью в направлении героя,

― стратегия предугадывания движения героя.

Результаты этих стратегий можно увидеть в приложениях. В первой стратегии ([видео 1 в приложении 1](https://docs.google.com/document/d/1aOkmYoR4Js61p6iopHsheMixIrI6_fHvy2t86fyaos4/edit?tab=t.0#heading=h.2tckb9k8ob4o)) враг проигрывает, а во второй - выигрывает. Во второй стратегии ([видео 2 в приложении 2](https://docs.google.com/document/d/1aOkmYoR4Js61p6iopHsheMixIrI6_fHvy2t86fyaos4/edit?tab=t.0#heading=h.da2o4gfy198m)) враг начинает преследовать героя. Как описано в стратегии героя, герой ускоряется при приближении врага. Враг не может догнать героя. Это связано с выбором начальной позиции. Враг попал на окружность при приближении к герою. То есть направление движения героя было практически 180 градусов по отношению к врагу. В стратегии врага предусмотрена передышка. Это значит, что когда силы врага кончаются, он останавливается, и ждет, пока накопится энергия (health). Это видно на видео. После передышки он заходит с другого угла и догоняет героя.

Таким образом видно что стратегию врага можно улучшить. Он может более выгодно подобрать угол. Соответственно стратегию героя тоже можно улучшить.  Например, герой может менять направление на противоположное, при приближении врага.

Необходимо отметить небольшую недоработку в стратегии героя. Ввиду неточного выбора поправки к углу наклона - круг по которому движется герой сужается.

Мы провели несколько несколько запусков с различными положениями врага и героя и выявили что при комбинации этих двух стратегий враг всегда побеждает.

Ниже приведены общие параметры для героя и врага.

|  |  |  |
| --- | --- | --- |
| Имя параметра | Значение для Врага | Значение для Героя |
| Максимальное здоровье | 900 | 900 |
| Максимальная скорость | 15 | 20 |
| Размеры поля | 20x20 | |
| Формула изменения здоровья | 5 - скорость | 5 - скорость + 1 |

Таблица 1. Значение параметров для героя и врага. В этой таблице задаются параметры для героя, врага и общие параметры.

## Выводы по второй главе

1. Разработана среда, в которой можно реализовать стратегии поведения персонажей (героя и врага) и увидеть результат работы этих стратегий.
2. Разработаны стратегии, в которых на каждом шаге автоматически определяется скорость и угол наклона героя и врага.
3. Проанализирована работа двух стратегий (движение по кругу и забегание вперед). В результате этого выявлено что стратегия забегания вперед выигрывает.
4. Я ознакомился с основами программирования на C#, Godot и проектирования программ.

# ЗАКЛЮЧЕНИЕ

В результате работы мы проделали следующее. Создали математическую модель игры, в которой враг догоняет героя и модель стратегий. Разработали среду в Godot, в которой можно реализовать стратегии героя и врага. Герой должен убежать от врага а враг должен его догнать. Стратегии автоматические. Они заключаются в том, что на каждом автоматически, с помощью алгоритма выбирается скорость и направление движения.

Мы проанализировали две стратегии и выяснили, что стратегия врага лучше. Следует отметить, что глобальные параметры, такие как максимальное здоровье, прирост здоровья и максимальную скорость можно менять. С помощью этого можно добиться баланса сил героя и врага. То есть, чтобы не было ситуации, что враг всегда побеждает. Если сбалансировать параметры, то шансы героя и врага на победу будут равны и выигрыш будет достигаться именно за счет стратегии, то есть за именно за счет интеллекта.

Мы проанализировали работу искусственного интеллекта и выявили что более высокоинтеллектуальные стратегии выигрывают у более простых стратегий.

# СПИСОК ЛИТЕРАТУРЫ

1. Интеллект. Большой психологический словарь [Электронный ресурс]/ Б.Г. Мещеряков, В. П. Зинченко. 2003 г ― Режим доступа: <https://psychological.slovaronline.com/> Дата обращения: 5.01.2025
2. Захаров А. Искусственный интеллект. Calltouch blog [Электронный ресурс] /Захаров А. – Режим доступа:  <https://www.calltouch.ru/blog/glossary/iskusstvennyj-intellekt/>  Дата обращения: 10.01.2025
3. Искусственный интеллект между миром и реальностью. Курьер ЮНЕСКО  – Режим доступа: <https://courier.unesco.org/ru/articles/iskusstvennyy-intellekt-mezhdu-mifom-i-realnostyu> Дата обращения: 8.01.2025
4. Что такое Big Data и почему их называют «новой нефтью». РБК – Режим доступа: <https://trends.rbc.ru/trends/innovation/5d6c020b9a7947a740fea65c?from=copy> Дата обращения: 8.01.2025
5. Интеллект. Толковый словарь Ожегова и Шведовой  [Электронный ресурс]: <https://gufo.me/dict/ozhegov/>  Дата обращения: 5.01.2025
6. Развитие искусственного интеллекта. Geek Brains – Режим доступа: <https://gb.ru/blog/razvitie-iskusstvennogo-intellekta/> Дата обращения: 6.01.2025
7. Среда программирования. Godot Engine ― Режим доступа.: <https://godotengine.org/> Дата обращения: 27.12.24

Oracle – Режим доступа: <https://www.oracle.com/cis/artificial-intelligence/machine-learning/what-is-deep-learning>

# ПРИЛОЖЕНИЯ

## Приложение 1. Видео 1 с визуализацией стратегий.

<https://drive.google.com/file/d/176VkO1ybIQfvBQOehIJCRrZs48Yev66L>

Стратегия врага: Движение с постоянной скоростью в направлении героя

Стратегия героя: Движение по кругу.

Враг движется с постоянной скоростью в направлении героя и не может его догнать.

## Приложение 2. Видео 2 с визуализацией стратегий.

<https://drive.google.com/file/d/1rmb37BHki5NSSLZgB-UPz8qpH2JcDPFY>

Стратегия врага: Стратегия предугадывания движения героя.

Стратегия героя: Движение по кругу.

Враг начинает преследовать героя. Как описано в стратегии героя, герой ускоряется при приближении врага. Враг не может догнать героя. Это связано с выбором начальной позиции. Враг попал на окружность при приближении к герою. То есть направление движения героя было практически 180 градусов по отношению к врагу. В стратегии врага предусмотрена передышка.

## Приложение 3. Ссылка на исходный код

<https://github.com/evgenycheryomushkin/Proekt>

Приложение 4.

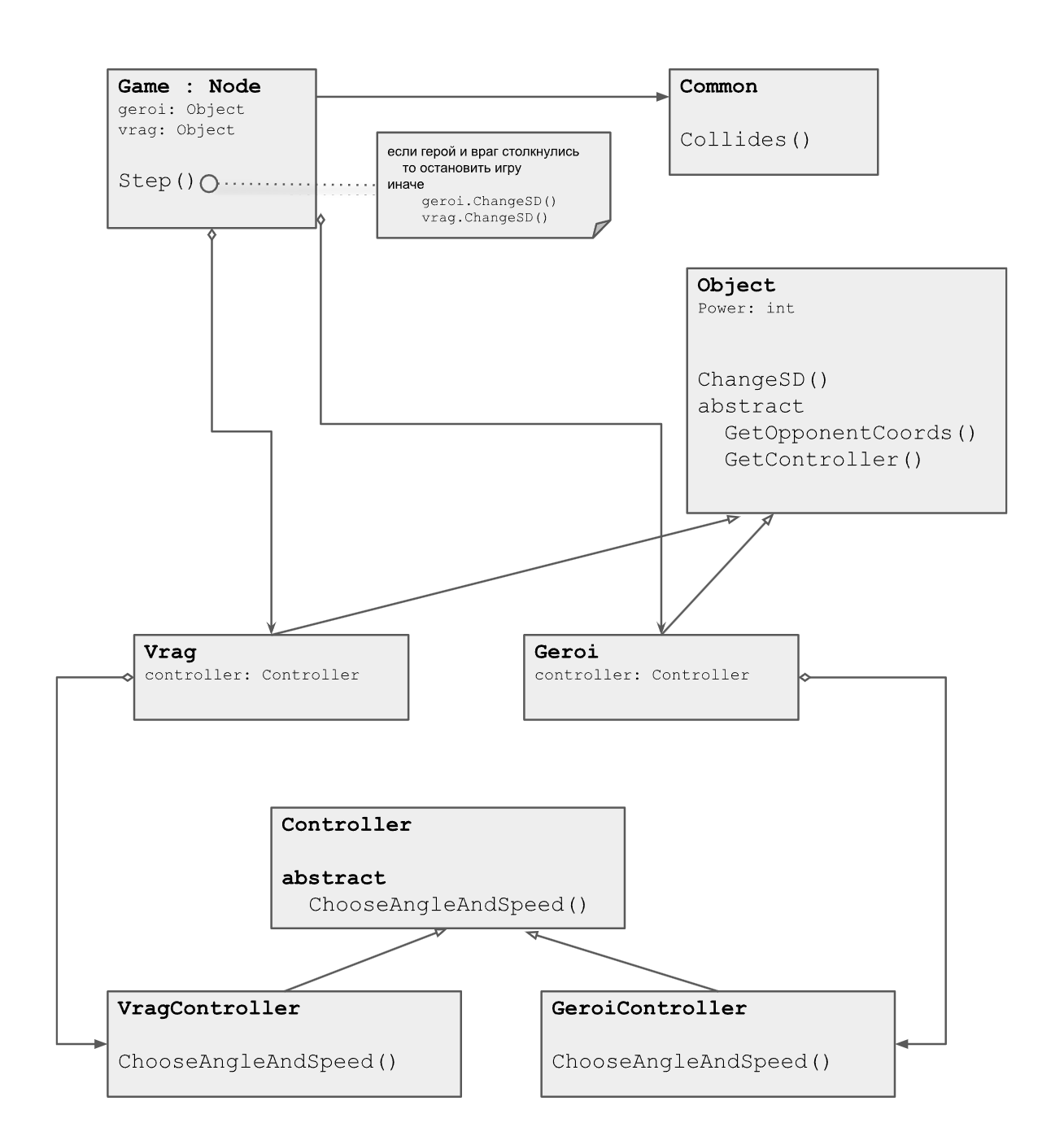


Рис 4. На данном рисунке изображена диаграмма классов. В классах - контроллерах (внизу) реализована логика поведения персонажей (стратегии). Она отделена от функционала игры, поэтому стратегии можно менять. У нас реализовано несколько стратегий (контроллеров) для врага и героя. Остальные классы (вверху) реализуют логику игры. Класс Game прикреплен к корневому узлу Godot. В классе Game раз в 100 миллисекунд вызываются основные функции пересчета скорости и направления. В классе Geroi и Vrag инициализируются контроллеры.

Основной класс нашей программы это класс Game. В нем объявлен шаг, с которым мы будем вызывать функции ChangeSpeedAndDirection, пересчитывающие скорость и угол наклона. Шаг задан в 100 миллисекунд. Объявлена функция \_Ready(), вызывающаяся один раз при старте. В ней запоминается начальное время. В функции \_PhysicsProcess(double delta) вычисляется номер шага с помощью формулы

Tick = ((int)(time - GameStartTimeMs)) / SHAG;

Далее, так как функция \_PhysicsProcess вызывается чаще чем через 100 миллисекунд, то проверяется, если номер шага изменился, то вызвать функцию Step(). Функция Step() вызывается ровно один раз за шаг. (Один раз за 100 миллисекунд). Соответственно в функции Step() реализована проверка на столкновение объектов и пересчет угла наклона и скорости.

Класс Common содержит функцию проверки на столкновение. Классы Vrag и Geroi наследуются от одного класса Object, так как у них много похожего функционала. Класс Object содержит функцию ChangeSpeedAndDirection. Она вызывается из Game. В ней происходит вызов

GetController().ChooseAngleAndSpeed(x, y, Rotation, opponentX, opponentY, Power, Tick);

где GetController() - абстрактная функция, реализованная по разному у Geroi и Vrag. У них разные контроллеры. Контроллер это сущность, которая вычисляет новые угол и скорость. Соответственно в вызове, указанном ранее мы вызываем функцию пересчета.

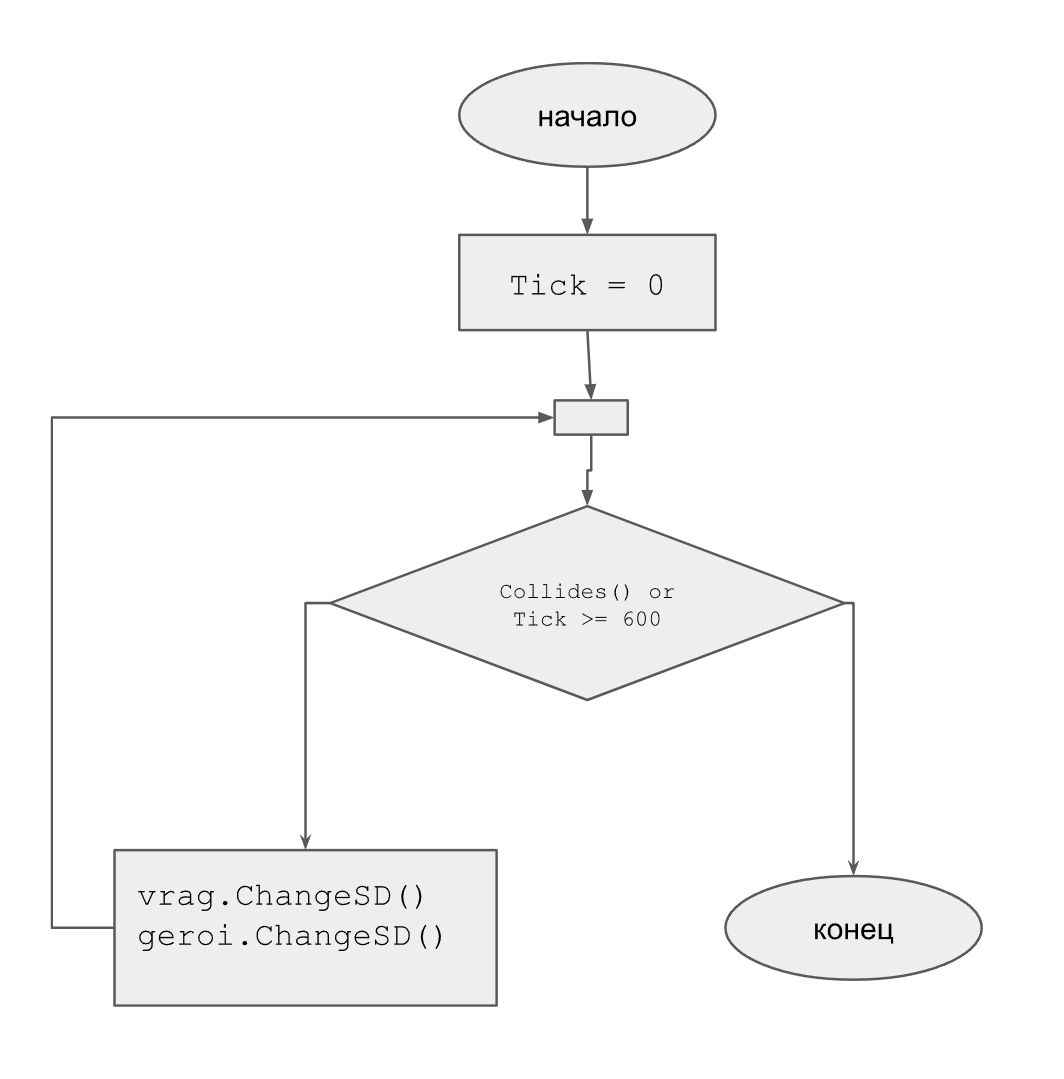


Рис 5. Блок схема выполнения программы.