УДК 004.9

Создание игры Breakout на игровом движке Godot

Черкашин Александр Михайлович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описан процесс создания игры Breakout в жанре аркада на игровом движке Godot. В процессе работы использовалась программа Godot для создания графического интерфейса (игровые объекты, игрок, мяч, кирпичи), и написан код программы на языке GDScript. Результат работы - игра Breakout на игровом движке Godot.
Ключевые слова: Godot, компьютерная игра, Breakout, аркадная игра.

Creation of Breakout game with the Godot game engine

Cherkashin Alexander Mihailovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of creating the game Breakout in the arcade genre on the Godot game engine. In the course of work, the Godot program was used to create a graphical interface (game objects, player, ball, bricks), and the program code was written in the GDScript language. The result of the work is the game Breakout on the Godot game engine.
Keywords: Godot, computer game, Breakout, arcade game.

1 Введение

1.1 Актуальность исследования

Данная статья описывает возможность создания игры Breakout на игровом движке Godot, в жанре аркада. Выбор игрового движка Godot для разработки игры обусловлен тем, что это один из самых легких и быстрых игровых движков под лицензией MIT (лицензия открытого и свободного программного обеспечения), который разрабатывается сообществом Godot Engine Community.

1.2 Цель исследования

Целью работы является разработка игры Breakout в жанре аркада.

1.3 Обзор исследований

В работе К.Е.Дилл, Й.Ц.Дилл определяют психологическую сложность и усиливающие характеристики образца самой популярной аркадной видеоигры, половые различия в игровом содержании и клиента, и описывает социальную структуру клиентов игровых автоматов, определяет

непредвиденные обстоятельства взаимодействия между людьми в играх, значение содержание игр с особым вниманием к насилию [1].

С.В.Дёминов описывает процесс разработки игры-платформера в среде Godot Engine [2].

С. Арномо разрабатывает и проектирует видеоигру платформера с использованием Godot Engine. Видеоигра называется «Охотник за монетами», в которой игроки управляют персонажем по имени Блан, чтобы собирать монеты, разбросанные в отдаленном лесу [3].

2. Рабочий процесс

В статье используется программа Godot версия 3.5.1.

Мы создали новый проект и назвали «Game_Breakout».

Компьютерная игра «Game_Breakout» представляет аркадную игру, где мяч перемещается до столкновения со стеной (края экрана) или кирпичом, и отскакивает от препятствия. Цель игры разбить все кирпичи, не допустив попадания мяча в пол, по достижению цели игра завершается (рис 2.2), если игрок не поймал мяч, мяч попадает в пол, игрок теряет жизнь.

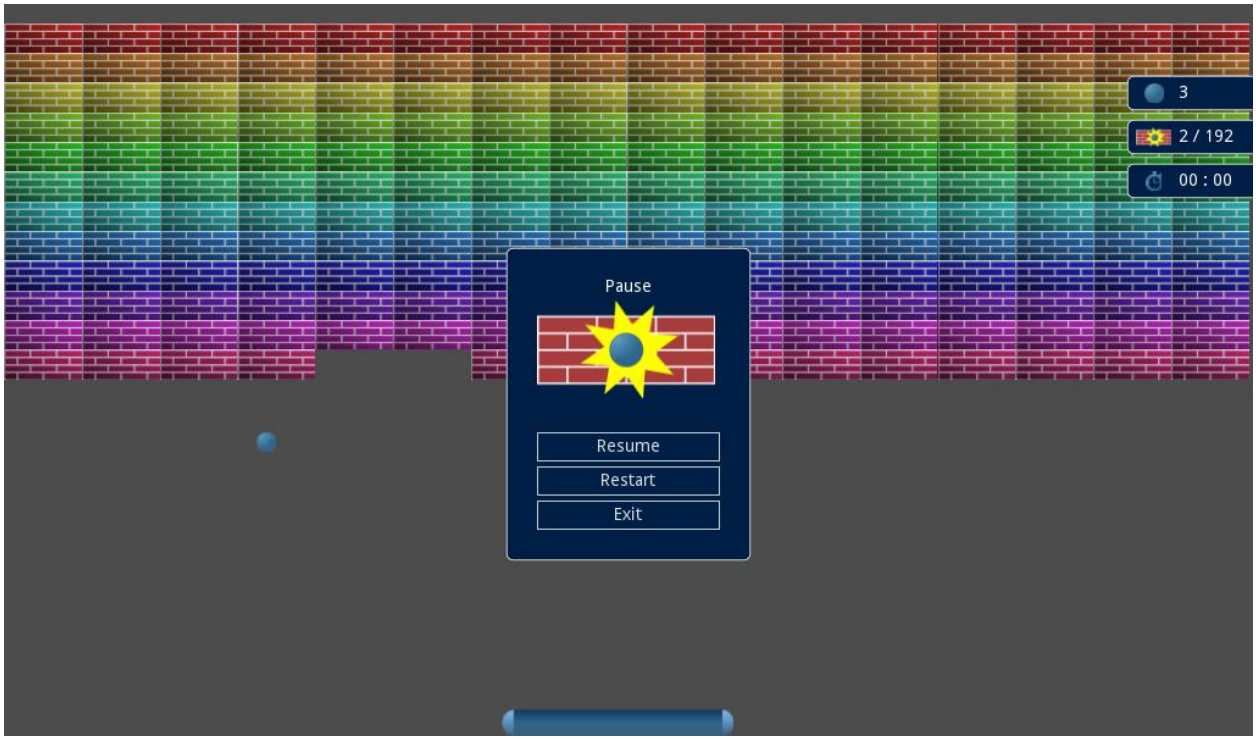

Рисунок 2.1. Игровой процесс

В правом углу экрана выводится оставшееся количество «жизней», время игры, количество разбитых/сколько осталось разбить кирпичей (рис 2.1).

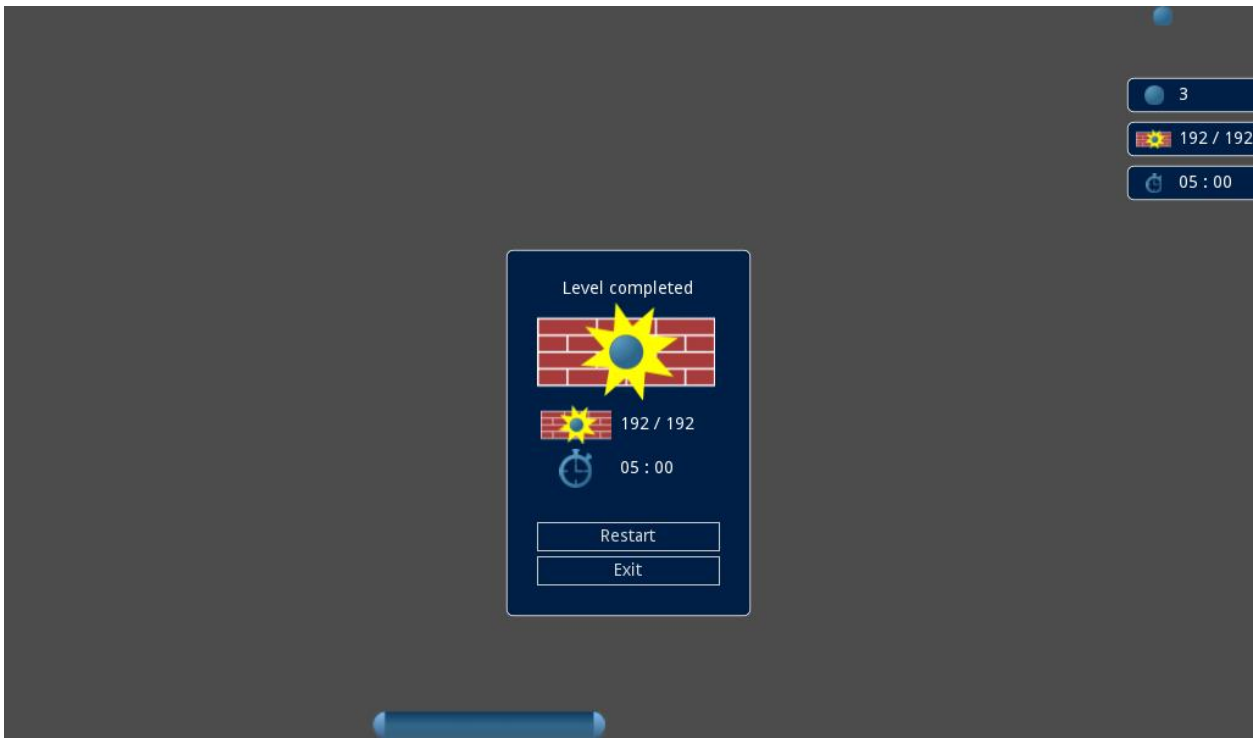

Рисунок 2.2. Игра закончилась с победы

В работе, мы изменили настройки проекта, «Имена слоя» → «2D Физика» (таблица 1), «Приложение» → «Запустить» в свойства «Главная сцена» установили значение «res://Level/Level_0.tscn». «Дисплей» → «Окно» в свойства «Попиксельная прозрачность» установлено «Разрешено» на Вкл.

«Дисплей» → «Окно» → «Растяжение» → «Режим» установлено на 2d.

Таблица 1. Имена слоя для 2D Физика

Название	Имя
Layer 1	Static
Layer 2	Kinematic

Таблица 2. Файловая структура данных, для игрового проекта

Путь	Описание
/.import	Каталог файлы импорта
/Asset	Каталог контента
/Asset/Texture	Каталог текстуры
/Asset/Texture/Ball.png	Рисунок «мяч»
/Asset/Texture/Brick.png	Рисунок «кирпич»
/Asset/Texture/Brick_broken.png	Рисунок «разбитый кирпич»
/Asset/Texture/Clock.png	Рисунок «часы»
/Asset/Texture/Paddle.png	Рисунок «весло»

/icon.png	Значок приложения
/Level	Каталог сцены
/Level/Level_0.tscn	Главная сцена
/Object	Каталог объекта
/Object/Brick_Red.tscn	Объект «кирпич»
/project.godot	Файл проекта
/Script	Каталог скрипта
/Script/Ball.gd	Скрипт «мяч»
/Script/Brick.gd	Скрипт «кирпич»
/Script/Hud.gd	Скрипт «Hud»
/Script/Level_0.gd	Скрипт для главный сцена
/Script/Player.gd	Скрипт «Игрок»

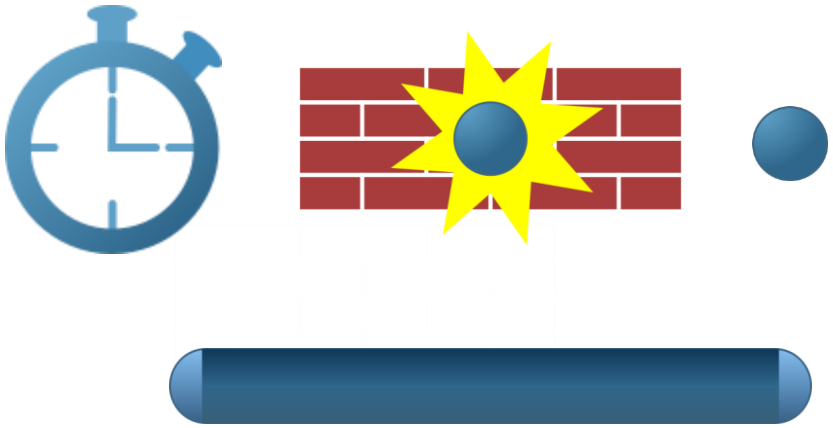

Рисунок 2.3Игровой контент «часы», «разбитый кирпич», «мяч», «весло»

Разработали контент (рис 2.3), используя программу Inkscape.

Создали 2D сцену и назвали «Level_0» и сохранили файл «/Level/Level_0.tscn» (рис 2.4).

В значок приложения установили рисунок «разбитый кирпич».

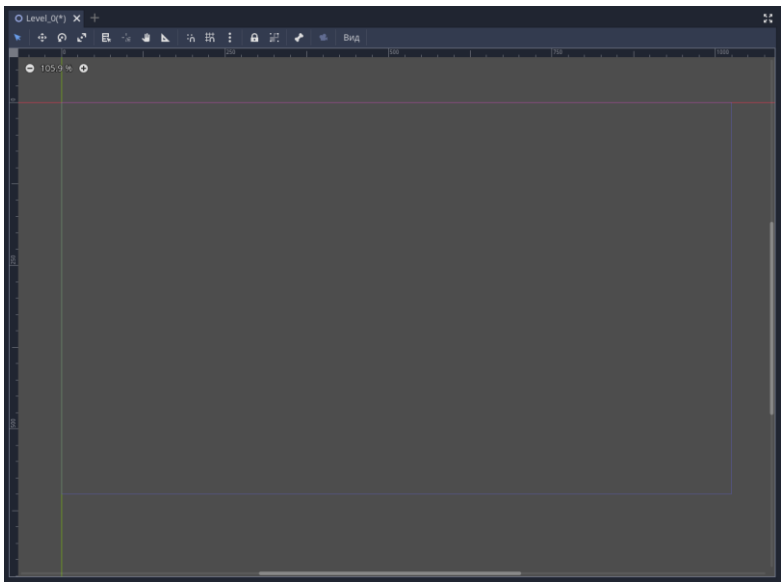

Рисунок 2.4 Пустая 2D сцена

В сцене «Level_0» добавили множество объектов разных типов и расположили по порядку (рис 2.5), и в (таблице 3) представлен полный список добавленных объектов в сцене «Level_0».

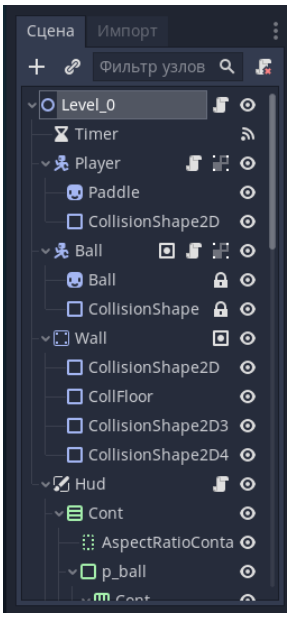

Рисунок 2.5 Структура сцены «Level_0»

Таблица 3. Структура сцены «Level_0»

Путь	Тип	Описание
Level_0	Node2D	Корневая сцена
/Timer	Timer	Таймер
/Player	KinematicBody2D	Игрок
/Player/Paddle	Sprite	Рисунок «весло»
/Player/CollisionShape2D	CollisionShape2D	Область столкновения

/Ball	KinematicBody2D	Мяч
/Ball/Ball	Sprite	Рисунок «Мяч»
/Ball/CollisionShape2D	CollisionShape2D	Область столкновения («Мяч»)
/Wall	StaticBody2D	Стена
/Wall/CollisionShape2D	CollisionShape2D	Область столкновения (левая стена)
/Wall/CollFloor	CollisionShape2D	Область столкновения (нижняя стена)
/Wall/CollisionShape2D3	CollisionShape2D	Область столкновения (правая стена)
/Wall/CollisionShape2D4	CollisionShape2D	Область столкновения (верхняя стена)
/Hud	CanvasLayer	Интерфейс «Hud»
/Hud/Cont	VBoxContainer	Вертикальное выравнивание
/Hud/Cont/AspectRatioContainer	AspectRatioContainer	Пробел
/Hud/Cont/p_ball	PanelContainer	Панель контейнера
/Hud/Cont/p_ball/Cont	HBoxContainer	Горизонтальное выравнивание
/Hud/Cont/p_ball/Cont/Tex	TextureRect	Область рисунка
/Hud/Cont/p_ball/Cont/Label	Label	Область текста «Мяч» для отчета количество доступных «Мяч».
/Hud/Cont/p_ball/Cont/AspectRatioContainer	AspectRatioContainer	Пробел
/Hud/Cont/AspectRatioContainer2	AspectRatioContainer	Пробел
/Hud/Cont/p_brickBroken	PanelContainer	Панель контейнера
/Hud/Cont/p_brickBroken/Cont	HBoxContainer	Горизонтальный выравнивания
/Hud/Cont/p_brickBroken/Cont/Tex	TextureRect	Область рисунка
/Hud/Cont/p_brickBroken/Cont/Label	Label	Область текста «разбитый кирпич».
/Hud/Cont/p_brickBroken/C	AspectRatioContainer	Пробел

ont/AspectRatioContainer		
/Hud/Cont/AspectRatioContainer3	AspectRatioContainer	Пробел
/Hud/Cont/p_time	PanelContainer	Панель контейнера
/Hud/Cont/p_time/Cont	HBoxContainer	Горизонтальное выравнивание
/Hud/Cont/p_time/Cont/Tex	TextureRect	Область рисунка
/Hud/Cont/p_time/Cont/Label	Label	Область текста «время игры».
/Hud/Cont/p_time/Cont/AspectRatioContainer	AspectRatioContainer	Пробел
/Hud/PanelContainer	PanelContainer	Окно
/Hud/PanelContainer/GridContainer	GridContainer	Таблица
/Hud/PanelContainer/GridContainer/AspectRatioContainer	AspectRatioContainer	Пробел
/Hud/PanelContainer/GridContainer/AspectRatioContainer2	AspectRatioContainer	Пробел
/Hud/PanelContainer/GridContainer/AspectRatioContainer3	AspectRatioContainer	Пробел
/Hud/PanelContainer/GridContainer/AspectRatioContainer4	AspectRatioContainer	Пробел
/Hud/PanelContainer/GridContainer/VBoxContainer	VBoxContainer	Вертикальное выравнивание
/Hud/PanelContainer/GridContainer/VBoxContainer/Label	Label	Заголовок
/Hud/PanelContainer/GridContainer/VBoxContainer/TextureRect	TextureRect	Рисунок подзаголовка
/Hud/PanelContainer/GridContainer/VBoxContainer/p_brickBroken	HBoxContainer	Горизонтальное выравнивание
/Hud/PanelContainer/GridCo	TextureRect	Рисунок «разбитый

ntainer/VBoxContainer/p_brickBroken/Tex		кирпич»
/Hud/PanelContainer/GridContainer/VBoxContainer/p_brickBroken/Label	Label	Текстовое поле, счетчик «разбитых кирпичей».
/Hud/PanelContainer/GridContainer/VBoxContainer/p_time	HBoxContainer	Горизонтальное выравнивание
/Hud/PanelContainer/GridContainer/VBoxContainer/p_time/Tex	TextureRect	Рисунок «Время игры»
/Hud/PanelContainer/GridContainer/VBoxContainer/p_time/Label	Label	Текстовое поле, «Время игры».
/Hud/PanelContainer/GridContainer/VBoxContainer/AspectRatioContainer6	AspectRatioContainer	Пробел
/Hud/PanelContainer/GridContainer/VBoxContainer/b_resume	Button	Кнопка «Продолжить игру».
/Hud/PanelContainer/GridContainer/VBoxContainer/b_restart	Button	Кнопка «Перезапустить игру».
/Hud/PanelContainer/GridContainer/VBoxContainer/b_exit	Button	Кнопка «Выход на рабочей стол».
/Hud/PanelContainer/GridContainer/AspectRatioContainer6	AspectRatioContainer	Пробел
/Hud/PanelContainer/GridContainer/AspectRatioContainer7	AspectRatioContainer	Пробел
/Hud/PanelContainer/GridContainer/AspectRatioContainer8	AspectRatioContainer	Пробел
/Hud/PanelContainer/GridContainer/AspectRatioContainer5	AspectRatioContainer	Пробел

Листинг 2.1. Скрипт «Level_0.gd» закрепленный объект «Level_0»

```
extends Node2D

export var balls = 3;
export var brickBroken = 0;
var timeGame = 0;
var _status_game = 0
const GAME_PLAY = 0
const GAME_OVER = 1
const GAME_WIN = 2

export var el_brick = preload("res://Object/Brick_Red.tscn");
export var gen:Vector2 = Vector2(16, 12)
export var bricks:int = 0; #16 * 12;
func _ready():
	var b_size:Vector2;
	var coord:Vector2;
	var brick:Node2D
	var tex:Texture;
	var tex_gradient:Gradient;

	for i in range(gen.x):
		for j in range(gen.y):
			self.bricks += 1;
			brick = el_brick.instance()
			brick.scale /= 2.0;
			brick.live = 1
			b_size = brick.get_node("Brick").get_rect().size * brick.scale

			coord.x = b_size.x * i * 2.0 + 32
			coord.y = b_size.y * j * 2.0 + 28;
			brick.position = coord;

			tex = brick.get_node("Background").texture.duplicate()
			tex_gradient = tex.gradient.duplicate()
			tex.gradient = tex_gradient
			brick.get_node("Background").texture = tex

			tex_gradient.set_color(0, Color.from_hsv(float(j) / 12, 0.8, 0.8, 1.0))

			add_child(brick)

	self.update_hud()

func dec_defeat():
	if (self._status_game == GAME_PLAY):
		self._status_game = GAME_OVER
		$Hud/PanelContainer.visible = true
		self.get_tree().paused = true
		$Hud/PanelContainer/GridContainer/VBoxContainer/Label.text = "Game over"
		$Hud/PanelContainer/GridContainer/VBoxContainer/b_resume.visible = false

		$Hud/PanelContainer/GridContainer/VBoxContainer/p_brickBroken.visible = true
		$Hud/PanelContainer/GridContainer/VBoxContainer/p_time.visible = true

func dec_win():
	if (self._status_game == GAME_PLAY):
		self._status_game = GAME_WIN
		$Hud/PanelContainer.visible = true
		self.get_tree().paused = true
		$Hud/PanelContainer/GridContainer/VBoxContainer/Label.text = "Level completed"
		$Hud/PanelContainer/GridContainer/VBoxContainer/b_resume.visible = false
```

```
62
63                      $Hud/PanelContainer/GridContainer/VBoxContainer/p_brickBroken.visible = true
64                      $Hud/PanelContainer/GridContainer/VBoxContainer/p_time.visible = true
65
66  func update_hud():
67          $Hud/PanelContainer/GridContainer/VBoxContainer/p_brickBroken.visible = false
68          $Hud/PanelContainer/GridContainer/VBoxContainer/p_time.visible = false
69
70          if (self.balls >= 0):
71                  $Hud/Cont/p_ball/Cont/Label.text = str(self.balls);
72          $Hud/Cont/p_brickBroken/Cont/Label.text = str("{0} / {1}".format([self.brickBroken, self.bricks]));
73          $Hud/PanelContainer/GridContainer/VBoxContainer/p_brickBroken/Label.text                    =
74  $Hud/Cont/p_brickBroken/Cont/Label.text
75          if (self.balls < 0):
76                  self.dec_defeat()
77          if (self.brickBroken >= self.bricks):
78                  self.dec_win()
79
80  func b_resume_press():
81          get_tree().paused = false
82          $Hud/PanelContainer.visible = false
83
84  func b_restart_press():
85          get_tree().reload_current_scene()
86          get_tree().paused = false
87          $Hud/PanelContainer.visible = false
88
89  func b_exit_press():
90          get_tree().quit()
91
92  func Timer_update():
93          self.timeGame += 1;
94
95          var seconds = self.timeGame % 60
96          var minutes = self.timeGame % 3600 / 60
97          var str_elapsed = "%02d : %02d" % [minutes, seconds]
98
99          $Hud/Cont/p_time/Cont/Label.text = str_elapsed
100         $Hud/PanelContainer/GridContainer/VBoxContainer/p_time/Label.text = str_elapsed
```

Листинг 2.1. Главный обработчик сцены.

Строка 3. Переменная «balls», количество доступных «мячей».

Строка 4. Переменная «brickBroken», счетчик разбитых «кирпичей».

Строка 5. Переменная «timeGame», время продолжительности игры.

Строка 6. Переменная «_status_game», состояние игры.

Строка 7. Константа «GAME_PLAY», режим игры.

Строка 8. Константа «GAME_OVER», состояние поражения (конец игры).

Строка 9. Константа «GAME_WIN», состояние победы.

Строка 11. Загружает ресурс «/Object/Brick_Red.tscn» объект «кирпич».

Строка 12. Переменная «gen», количество по ячейкам для создания объекта «кирпич».

Строка 13. Переменная «bricks», количество объектов «кирпич» на сцене.

Строка 15 — 40. Создает объектов «кирпич» по ячейкам.

Строка 15. Переменная «b_size», размер объекта «кирпич».

Строка 16. Переменная «coord», расположение объекта «кирпич».

Строка 17. Переменная «brick», экземпляр объекта «кирпич».

Строка 18. Переменная «tex», текстура объекта «кирпич».

Строка 19. Переменная «tex_gradient», объектный градиент объекта «кирпич».

Строка 21 — 22. Итерация по ячейкам.

Строка 23. Суммирует количество созданных объектов «кирпич».

Строка 24. Создает экземпляр объекта «кирпич».

Строка 25. Уменьшает объекта «кирпич» в два раза.

Строка 26. Задаем переменной «live» значение 1 для объекта «кирпич».

Строка 27. Получаем размер объекта «кирпич».

Строка 29 - 31. Вычисляет расположение объекта «кирпич».

Строка 33 - 34. Делает связь объекта текстура и градиент уникальным для объекта «кирпич».

Строка 38. Изменяет цвет объекта «кирпич».

Строка 40. Добавляет экземпляр объекта «кирпич» в сцену.

Строка 44 и 55. Функция изменяет состояние игры. После изменение состояние игры на победу или поражение, скрывает кнопку «продолжить» (строка 50 и 61), отображает статистику (строка 52 — 53 и 63 — 64), изменяет текстовое поле (строка 49 и 60), устанавливает паузу (строка 48 и 59), отображает окно меню (строка 47 и 58), изменяет переменную «_status_game» (строка 46 и 57).

Строка 66 - 73. Обновляет интерфейс Hud.

Строка 74 — 77. Проверяет условия игры, и при выполнения условия, изменяет состояние игры.

Строка 79 — 81. Кнопка «продолжить», продолжает игру.

Строка 83 — 86. Кнопка «Перезапустить», перезапускает игру.

Строка 88 — 89. Кнопка «выход», завершает игру, выходит на рабочий стол.

Строка 91 — 99. Считает время игры в секундах, выводит в формате мм:сс.

В таблице 3, объекты «/Ball» и «/Wall» добавили в группу «Ball» и «Wall».

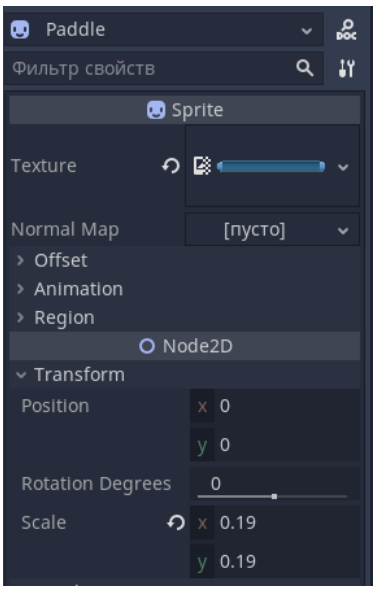

Рисунок 2.6 Панель свойств для объекта «/Player/Paddle»

Объекты «/Player/Paddle» и «/Ball/Ball», размером «Scale» были заданы с координатами (x: 0.189655, y: 0.189655) и (x: 0.137931, y: 0.137931). Текстура «весло» и «мяч» (рис 2.6).

У обоих объектов «/Player» и «/Ball» был задан слой столкновения «collision», «layer», «mask» 1 и 2 включительно. Заданы координаты «Transform → Position» (x: 488, y: 576) и (x: 504, y: 416).

Объекту «/Timer» задано значение «autostart» на True, и установлен сигнал «timeout» → к объекту «Level_0» в методе «Timer_update».

Мы нарисовали границы в краях экрана, объект «/Wall» для предотвращения вылетов объекта «мяч» за пределы пространства экрана.

Листинг 2.2. Скрипт «Ball.gd» закрепленный объект «/Ball»

```
1   extends KinematicBody2D
2
3   export var speed:Vector2 = Vector2(350, 200);
4   export var dir_speed:Vector2 = Vector2(-1, -1);
5   export var acceleration:float = 8;
6   export var vel:Vector2 = Vector2(0, 0);
7   export var snap:Vector2 = Vector2.ZERO;
8   export var dir_up:Vector2 = Vector2.UP
9   export var stop_on_slope:bool = false
10  export var floor_max_angle:float = deg2rad(45.0)
11  export var def_pos:Vector2;
12
13  func _ready():
14          self.def_pos.x = ProjectSettings.get_setting("display/window/size/width") / 2
15          self.def_pos.y = 416
16
17  func _physics_process(delta):
18          vel.x = vel.linear_interpolate(dir_speed * speed, acceleration * delta).x
19          vel.y = vel.linear_interpolate(dir_speed * speed, acceleration * delta).y
20
21          vel = move_and_slide_with_snap(vel, snap, dir_up,
22                          stop_on_slope, 4, floor_max_angle, false)
23
24          var a:float
25          var coll:KinematicCollision2D
26          var c:Vector2
27          for i in get_slide_count():
28                  coll = get_slide_collision(i)
29                  c = coll.normal.round();
30                  if coll.collider && coll.collider is RigidBody2D:
31                          coll.collider.apply_central_impulse(-coll.normal * vel.length() / 100.0)
32                  if c.x != 0:
33                          dir_speed.x = c.x;
34                  if c.y != 0:
35                          dir_speed.y = c.y;
36
37                  if (coll.collider.name == "Player"):
38                          a = (coll.position - coll.collider.position).x;
39                          if a < 0:
40                                  a = a - ceil(a);
41                          else:
42                                  a = a - floor(a);
43                          dir_speed.x = sign(a);
44                          dir_speed.y -= abs(a);
45
46                  if (coll.collider.is_in_group("Wall") and coll.collider_shape.name == "CollFloor"):
47                          self.dir_speed = Vector2(0, 0);
48
49                          self.visible = false
50                          self.position = self.def_pos;
51                          get_viewport().get_node("Level_0").balls -= 1;
52                          get_viewport().get_node("Level_0").update_hud();
53
54                          yield(get_tree().create_timer(1.0), "timeout")
55                          self.visible = true
56                          self.dir_speed = Vector2(-1, -1);
57                          break;
```

Листинг 2.2. Переменные представлены в таблице 4, в строках 3 — 11.

Таблица 4. Перемены для листинга 2.2

Название	Тип	Значение	Описание
speed	Vector2	(350, 200)	Максимальная скорость
dir_speed	Vector2	(-1, -1)	Направление скорости
acceleration	float	8	Ускорение
vel	Vector2	(0, 0)	Текущая скорость
snap	Vector2	(0, 0)	Вектор привязки к земле
dir_up	Vector2	(0, 1)	Направление вектора вверх
stop_on_slope	bool	false	Стоять по склонам.
floor_max_angle	float	deg2rad(45.0)	Максимальный угол при котором тело считается полом или потолком.
def_pos	Vector2	(0, 0)	Расположение тела по умолчанию.

Строки 14 — 15 устанавливают «def_pos» начальное значение, расположение для объекта «мяч».

Строки 18 — 19 вычисляют линейную интерполяцию «vel» между dir_speed * speed относительно acceleration * delta, полученное вычисление изменяет переменную vel.

Строка 21 перемещает тело «мяч» с заданной скоростью «vel» по различным склонам.

Строка 27 перебирает список столкнувших тел.

Строка 29 получаем вектор направления столкновения тела «мяч».

Строка 31 «мяч» толкает столкнувшееся тело.

Строки 32 — 35 отражают скорость направления «мяч» после столкновения.

Строка 37 проверяет объект столкновения «весло».

Строка 38 вычитает «coll.position» (точка столкновение) между «coll.collider.position» (расположение объекта «весло») по осью x.

Рисунок 2.7. Направления движений «мяч» после столкновения с объектом «весло»

Строки 39 — 42 переменная «a» получает значение от 0 до 1.

Строки 43 — 44 вычисляют направление движения объекта «мяч» (рис 2.7).

Строка 46 определяет столкновение объекта «мяч» со стеной и определяет является ли стена полом.

Строки 47 — 57 после столкновения объекта «мяч» с полом, вычитает количество оставшихся «мячей».

Листинг 2.3. Скрипт «Player.gd» закрепленный объект «/Player»

```
1   extends KinematicBody2D
2
3   export var gravity_force := 100.0
4   export onready var gravity = (ProjectSettings.get_setting("physics/2d/default_gravity") * gravity_force)
5   export onready var gravity_dir = (ProjectSettings.get_setting("physics/2d/default_gravity_vector") * gravity)
6
7   export var speed:Vector2 = Vector2(500, 400);
8   export var dir_speed:Vector2 = Vector2(0, 0);
9   export var acceleration:float = 8;
10  export var vel:Vector2 = Vector2(0, 0);
11  export var snap:Vector2 = Vector2.ZERO;
12  export var dir_up:Vector2 = Vector2.UP
13  export var stop_on_slope:bool = false
14  export var floor_max_angle:float = deg2rad(45.0)
15
16  export var mouse_pos:Vector2 = Vector2(0.0, 0.0)
17  func _ready():
18          self.mouse_pos.x = ProjectSettings.get_setting("display/window/size/width") / 2
19
20  func _input(event):
21          if event is InputEventMouseMotion:
22                  self.mouse_pos = event.position
23
24  func _physics_process(delta):
25          var pos_x = mouse_pos.x - self.position.x
26          speed.x = max(10, abs(pos_x * 10.0));
27          dir_speed.x = sign(pos_x) * sign((max(20, abs(pos_x)) - 20) / 20)
28
29          if not is_on_floor():
30                  vel += gravity_dir * delta
31
32          vel.x = vel.linear_interpolate(dir_speed * speed, acceleration * delta).x
33          vel.y = vel.linear_interpolate(dir_speed * speed, acceleration * delta).y
34
35          vel = move_and_slide_with_snap(vel, snap, dir_up,
36                          stop_on_slope, 4, floor_max_angle, false)
```

Листинг 2.3. Переменные представлены в таблице 4 в строках 7 — 14.

Строка 16 переменная «mouse_pos» - координаты курсора.

Строки 20 — 22 получение расположения курсора.

Строки 25 — 27 вычисляют следование объекта «весло» за курсором по оси X.

Строки 29 — 30 вычисляют попадание объекта «маяч» в «весло».

Строки 32 — 33 вычисляют и получают скорость.

Строки 35 — 36 перемещают тело «весло» с заданной скоростью «vel».

Объекту «/Hud» заменено свойство «Pause Mode» на «Process».

Листинг 2.4. Скрипт «Hud.gd» закрепленный объект «/Hud»

```
1  extends CanvasLayer
2
3  func _input(event:InputEvent) -> void:
4          if event.is_action_pressed("ui_cancel") and $"..".status_game == $"..".GAME_PLAY:
5                  if ($"..".get_tree().paused):
6                          $"..".get_tree().paused = false
7                          $PanelContainer.visible = false
8                  else:
9                          $"..".get_tree().paused = true
10                         $PanelContainer.visible = true
```

Листинг 2.4. Выполняет паузу игры и открытия окна меню.

Строка 4 event.is_action_pressed проверяет нажатие кнопки клавиша Esc и сравнивает _status_game в константе GAME_PLAY.

Строка 5 проверяет состояние паузы в объекте «Level_0».

Строка 6 и 9 изменяет состояние паузы в объекте «Level_0»

Строка 7 и 10 изменяет состояние отображение окна меню «/Hud/PanelContainer».

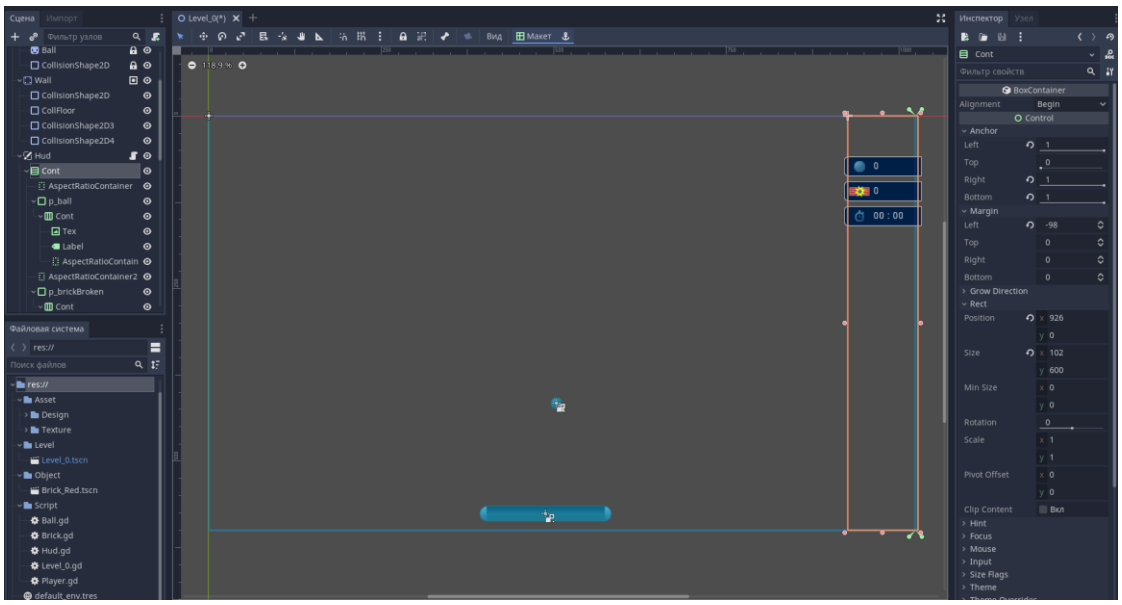

Рисунок 2.8. Панель свойства для объекта «/Hud/Cont»

Объекту «/Hud/Cont» задано свойства «Size», «Anchor», «Position» (Рис 2.7)

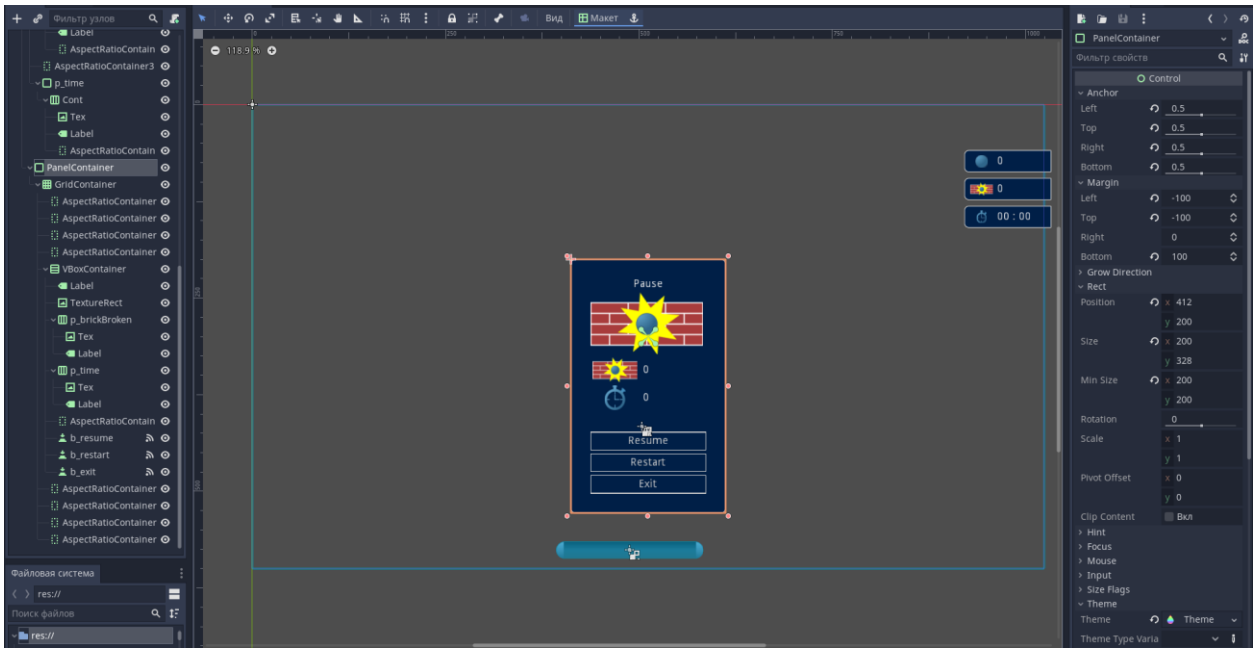

Рисунок 2.9. Панель свойств для объекта «/Hud/PanelContainer»

Объект «/Hud/PanelContainer/GridContainer/VboxContainer» содержит несколько дочерних объектов — 3 кнопки «продолжить», «перезапустить», «выход», все имеют связь сигналом вызова к объекту «Level_0».

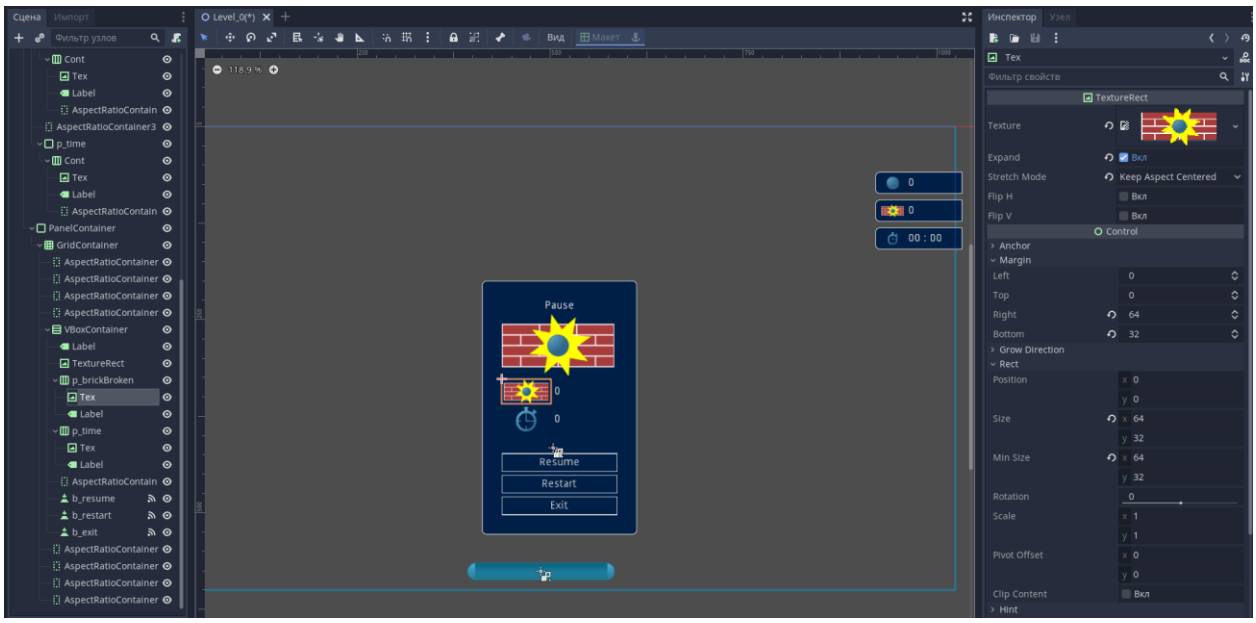

Рисунок 2.10. Выделенный объект «/Hud/PanelContainer/GridContainer/VBoxContainer/p_brickBroken/Tex»

Путь «/Hud/PanelContainer/GridContainer/VboxContainer» содержит счетчики, конечный результат завершенный игры, 3 кнопки действия (рис 2.9).

Объекту «Label» поля Text значение «0».

Путь «/Hud/Cont» содержит панель информации счетчиков справа экрана (рис 2.9).

Таблица 5. Структура сцены «Brick_Red»

Путь	Тип	Описание
Brick_Red	StaticBody2D	«Кирпич»
/Background	Sprite	Фон
/Brick	Sprite	Текстура «Кирпич»
/Light2D2	Light2D	Источник освещения
/CollisionShape2D	CollisionShape2D	Область столкновения
/Area	Area2D	Область
/Area/CollisionShape2D2	CollisionShape2D	Область столкновения

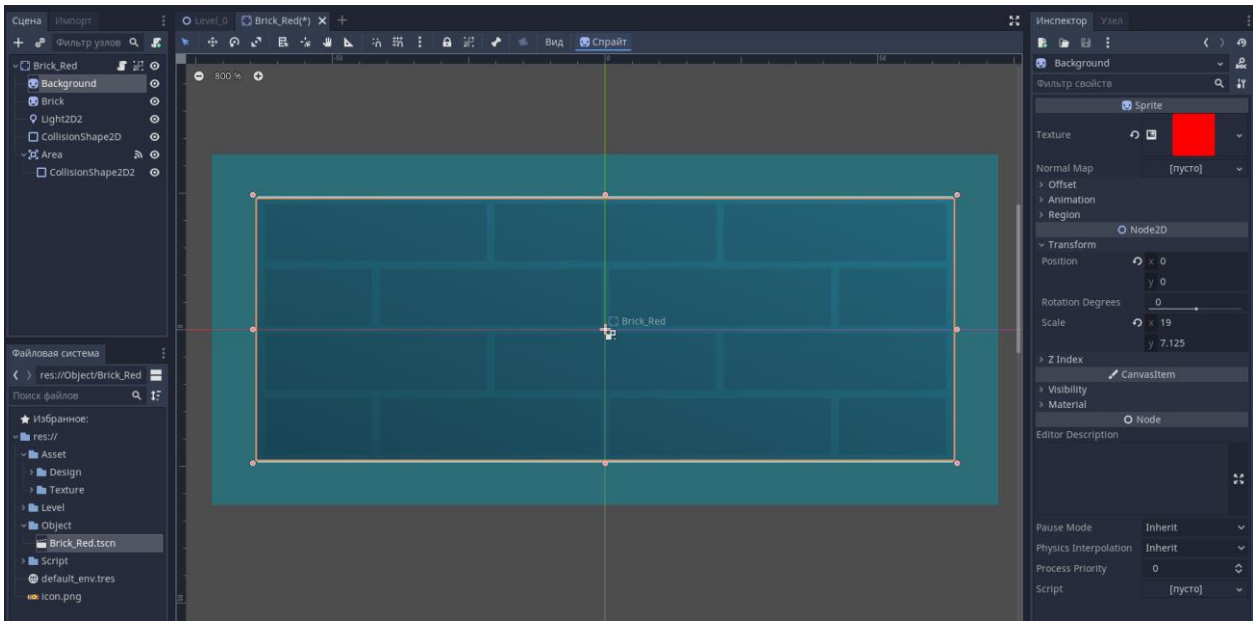

Рисунок 2.11. Изменение текстуры «кирпич»

Объект «/Brick» в свойства Texture задан путь к изображению объекта «кирпич», в Scale задано значение (x: 2, y: 2).

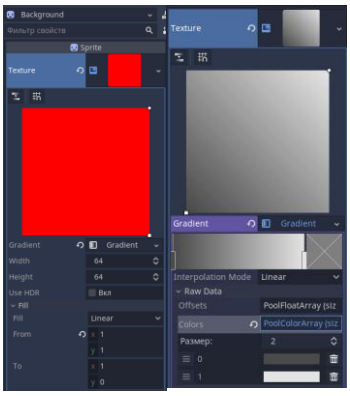

Рисунок 2.12. Заданно в свойстве значение градиента, цвет красный для объекта «/Background», а серый градиентный для объекта «Light2D2»

Объекту «/Light2D2» в свойстве Texture задан тип градиент, цвета для первой точки (0.292969, 0.292969, 0.292969, 1) — темный серый, для второй точки (0.894531, 0.894531, 0.894531, 1) — светлый серый (рис 2.11 справа). В панели «Light2D» в свойства «Mode» выбрано «Mix», Shadow → Enabled задано «Вкл». Scale задано (x: 19, y: 7.125).

Объект «/CollisionShape2D» в свойстве Shape выбран тип «RectangleShape2D» и задано Extents значение (608, 228). В объекте «/Area/CollisionShape2D2» свойству Extents задано значение (684.001, 304).

Объекту «/Area» задан сигнал «Area2D → body_entered», путь в «Brick_Red» вызываемый методом «Body_entered».

Объект «Brick_Red» имеет закрепленный скрипт «Brick.gd» (листинг 2.4).

Листинг 2.5. Скрипт «Brick.gd»

```
1  extends StaticBody2D
2
3  export var live = 3;
4
5  func coll_break(body):
6          self.live -= 1;
7          if (self.live <= 0):
8                  get_viewport().get_node("Level_0").brickBroken += 1;
9                  get_viewport().get_node("Level_0").update_hud();
10                 queue_free()
11         pass
12
13 func Body_entered(body):
14         if (body.is_in_group("Ball")):
15                 self.coll_break(body)
```

Листинг 2.5. Строка 3. Переменная live — количество жизни объекта «кирпич», после удара мяча вычитается live по достижения 0, «кирпич» удаляется.

Строка 13 функция Body_entered принимает сигнал от объекта «/Area», и определяет группу «Ball», вызывает функцию coll_break. Сигнал выполняется когда объект «мяч» сталкивается с объектом «кирпич».

Строка 5 функция «coll_break» выполняет действие разбитый «кирпич», вычитая live, проверяет достижения переменной live значения 0.

3 Выводы

В данной статье был описан ход разработки игры в жанре аркада Breakout. Разработан сценарий игры, подобраны рисунки, игровой контент и реализована программная составляющая игры.

Библиографический список

1. Dill K. E., Dill J. C. Video game violence: A review of the empirical literature // Aggression and violent behavior. 1998. Т. 3. №. 4. С. 407-428.
2. Дёминов С. В. Создание игры-платформера в среде Godot Engine //Материалы XXII Всероссийской научно-практической конференции молодых ученых, аспирантов и студентов, с международным участием в г. Нерюнгри, посвященной 30-летнему юбилею Технического института (филиала) СВФУ им. МК Аммосова. 2022. С. 220-225.
3. Arnomo S. et al. Perancangan game platformer pemburu koin menggunakan godot engine //Computer and Science Industrial Engineering (COMASIE). 2022. Т. 6. №. 4. С. 109-117.