

Министерство цифрового развития, связи и массовых коммуникаций РФ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Сибирский государственный университет телекоммуникаций и информатики»  
(СибГУТИ)

# КУРСОВАЯ РАБОТА

Вариант 15

Qt-3d-игровой движок и Тамагочи питомцы

Выполнил студент группы ИП-111 Прилепа М.К.

Проверил доцент кафедры ПМиК к.т.н Мерзлякова Е.Ю.

Новосибирск 2023

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	4
1. АНАЛИЗ ЗАДАЧ И ПОЛЬЗОВАТЕЛЕЙ .....	5
2. ВЫБОР РЕПРЕЗЕНТАТИВНЫХ ЗАДАЧ .....	7
3. ЗВИМСТВОВАНИЕ .....	8
Рисунок 1. Все виды животных в игре .....	8
Рисунок 2. Кот вблизи .....	8
Рисунок 3. 5 уровней отходов питомца в игре .....	9
Рисунок 4. Вид моделек в блендере .....	10
Рисунок 5. Вид 1-ой страницы отображения моделек ..	10
4. ПРОТОТИП ИНТЕРФЕЙСА .....	11
Рисунок 6. Вид стартового окна .....	11
Рисунок 7. Вид главного окна .....	12
Рисунок 8. Навигатор .....	13
Рисунок 9 и 10. Оригинальный вид навигатора .....	14
5. РЕАЛИЗАЦИЯ .....	15
Рисунок 11. Основные характеристики питомца .....	16
Рисунок 12. Нацеленный курсор на цель в виде отходов питомца .....	17
Рисунок 13. Нацеленный курсор на еду .....	18
Рисунок 14. Вид шейдера – выделителя цели вблизи ..	18
Рисунок 15. Цель – кровать .....	19
Рисунок 16. Техническое меню первого уровня маршрутизатора питомца .....	20
Рисунок 17. Техническое меню второго уровня маршрутизатора питомца .....	20



# ВВЕДЕНИЕ

Я выбрал данную тему по тому, что это, во-первых, игра, а во-вторых, я никогда не имел дела с Тамагочи питомцами до этой курсовой работы.

Изначально это была просто тема Тамагочи питомцев, но во время разработки этой темы я очень далеко ушёл в сторону разработки самих игровых движков. За основу был взят графический фреймворк Qt, обёрнутый в его собственную IDE “Qt Creator”. В данной IDE все разработки и велись.

Трудно сказать, какие приложения похожи на Тамагочи питомцев, но больше всего мне Тамагочи напомнила игру wobble dogs, в которой в качестве питомца выступают трёхмерные собачки с некоторым игровым генетическим рядом признаков. Вероятнее всего эта игра и побудила меня в данном курсовом проекте больше топить в сторону 3d, ежели неинтересного 2d. Я в последнее время вообще сторонюсь 2d игры, видимо по тому, что 3d лучше, ежели, чем в 2d. Да и в нём (в 3d) гораздо больше полёт фантазий у графических программистов, как в плане шейдерного программирования (GLSL), так и в плане создания самих трёхмерных моделек и построение трёхмерных сцен. Это интереснее на целое измерение!

В плане того, на что моё приложение похоже больше всего. На деле на 95% вышедших хороших 3d-игр, в которых действительно есть какой-то начинающий программист игровых движков. Я как-то смотрел ролик про современные этапы разработки современных игр и там действительно разработка игровых движков, т.е. некоторых велосипедов, действительно актуальная штука, а такое понятие, как “разработчик игровых движков” действительно настоящая профессия, а не просто какая-то вставка какого-то анекдота.

## 1. Анализ задач и пользователей

Разработка данного варианта задания «Qt-3d-игровой движок и Тамагочи» питомцы производилось полностью индивидуально без чьей-либо помощи.

В данной игре присутствуют профессиональные 3d-модели либо с, довольно-таки, высокой стоимостью, либо вообще без возможности купить их, но при этом без возможности скачать и использовать в коммерческих целях (здесь вернее именно запрет на коммерчески цели не автора). Разумеется, данные модели были не куплены, а взяты с сайта по продаже и демонстрации моделек “[sketchfab.com](http://sketchfab.com)”. Модели не совсем честно были бесплатно вынесены с данного сайта моей собственной программой на JavaScript (превышающей 3000 строк программного, практически некомментированного, кода) именуемой граббером. Конечно, сам факт использования моделей в учебных целях полностью законен, ибо курсовые проекты обычно не используются в коммерческих целях. Разумеется, разработку и сам программный код граббера в данной курсовой я не буду рассматривать в целях безопасности самого кода. Поверьте, что в интернете вы его скорее всего никогда не найдёте. Я пытался найти что-то подобное, но хакеры, что способны создать столь сложные системы взломов, умны, чтобы не выкладывать такое, а прочая часть хакеров ещё новички в своём деле.

Тем ни менее я могу перечислить пару студентов-друзей, что уже “держали” мою курсовую работу. Они были только тестирующими данной программы с основной целью, что работает не только у меня. До студентов так и не дошёл программный код, а только скомпилированный .exe результат, окружённый соответствующими Qt-библиотеками “.dll”, без которых вся эта скомпилированная “лапша” ассемблерного кода не будет работать. Обратить компиляцию с инструментами подобных масштабов без тяжёлого софта практически нереально.

Из данной категории (тестеры) были отобраны данные пользователи:

1. Евгений Флат: возраст – 20 лет, образование – среднее общее, студент, навыки в выбранной сфере – хороший друг и тестер, навыки владения компьютером – высокие.
2. Никита Патрушев: возраст – 20? лет, образование – среднее общее, студент, навыки в выбранной сфере – хороший друг, тестер и одноклассник, навыки владения компьютером – высокие.
3. Даниил Шпилев: возраст – 20 лет, образование – среднее общее, студент, навыки в выбранной сфере – хороший друг и тестер, навыки владения компьютером – высокие.

Пользователи из рассматриваемой области имеют навык владения компьютером обычно на высоком уровне. Они обладают лидерскими качествами и ожидают от приложения простой и четкой помощи в решении своих задач. Существенное значение имеет наглядность процесса управления.

Основные термины:

- **Габбер.** В общем смысле – это устройство/средство для извлечения какой-либо информации. В нашем же случае — это любая программа, которая вынимает из любого десктопного, браузерного и т.д. приложения какие-либо ресурсы деятельности человеческого творчества. Сюда входят и программный код, и музыка, и поэзия, и 3d-сетки, и текстуры от 3d-сеток, и просто текстуры – не для 3d-моделей. Список можно перечислять бесконечно. Конкретно в случае данной курсовой работы габбер направлен только на 3d-модели.

- **3d – трёхмерность.** Что-либо, имеющее три измерения. В данной ситуации речь идёт об Евклидовом пространстве.

- **Трёхмерная графика** – Раздел компьютерной графики, посвящённый методам создания изображений или видео путём моделирования объектов в трёх измерениях.

- **3d-моделирование** – процесс создания трёхмерной модели объекта.

- **3d-модель** – плод человеческого труда, состоящий из трёхмерной сетки полигонов и двухмерной карте ряда текстур. Текстуры могут быть, как и обычные/базовые, так и PBR.

- **PBR (от англ. physically based rendering)** – это метод наложения шейдеров и рендеринга, обеспечивающий взаимодействие света с материалом в соответствии с законами физики.

Возможности существующих программ по схожей тематике:

**Wobbledogs** — это симулятор жизни, напоминающий вариацию на тему игры Spore, только про выращивание собак из куколок.

Вот несколько особенностей игры:

1. **Существа, которых игрок выращивает в Wobbledogs, напоминают собак очень условно:** у них есть голова, уши, хвосты и лапы, но генетические маркеры могут лечь так, что получится что-то многолапое.
2. **Собаки постоянно мутируют**, причём делают это достаточно причудливым образом. В какой-то момент собака может окуклиться и провести некоторое время в коконе, прилепленном к потолку. Затем из куколки вылупится нечто прекрасное.
3. **Собаки в Wobbledogs физически смоделированы вплоть до внутренностей.** Мутации завязаны на пищу, которую они употребляют: в зависимости от корма и флоры, которую он привносит в собак, изменения могут пойти в разных направлениях.
4. **Собаки обладают уникальными квази-личностями**, определяющими их поведение при взаимодействии с объектами или другими животными.

Помимо выращивания собак, игроки смогут заняться проектированием миров-приютов для своих питомцев.

## 2. Выбор репрезентативных задач

1. Указание, куда питомцу идти. В основном это посе́вы с едой, отходы (чтобы их убрать) и кровать (чтобы поспать).
2. Выбор одного из 25 видов питомцев.
3. Задание имени питомцу.
4. Очистка переполненной отходами ячейки.
5. Использование специальных опций разработчика данного приложения, чтобы посмотреть состояния двухуровневого маршрутизатора питомца, взглянуть на то, как выглядит альтернативный цветовой мир, предназначенный для определения, на какую 3d-модельку направлен курсор мыши, или просто пролистать все модельки постранично, где страница – сетка 8 на 8.





- 4 модели скибиди-унитазов:

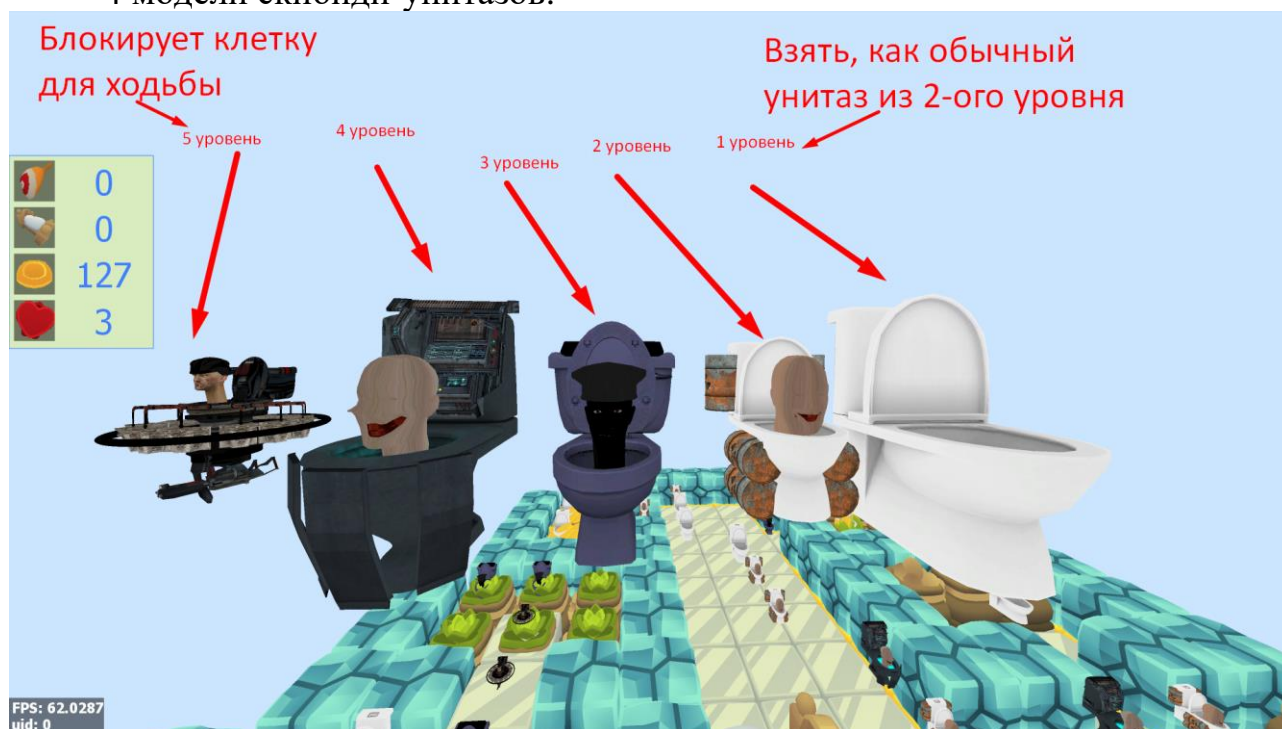


Рисунок 3. 5 уровней отходов жизнедеятельности питомца в игре.

Здесь отчётливо виден косяк с отображением текстур на моделях, но тем ни менее, сами сетки моделей в полном порядке. Пока ещё не разобрался, почему blender видит модели правильно, а моя программа показывает то, что действительно зашито в те .obj-файлы. Кстати, моя программа не понимает blender, поэтому используется экспорт сцены в .obj-файл. Моя программа и его не понимает, т.к. я разработал собственный формат .asd-моделей, который собирается из нескольких .obj-файлов с помощью отдельной моей Python3-программы.

- Гигантский набор из 808 моделей для построения миров  
<https://sketchfab.com/3d-models/cube-world-full-set-proto-series-47395bd7e2a14c2999cbb50d6fab7d51>:

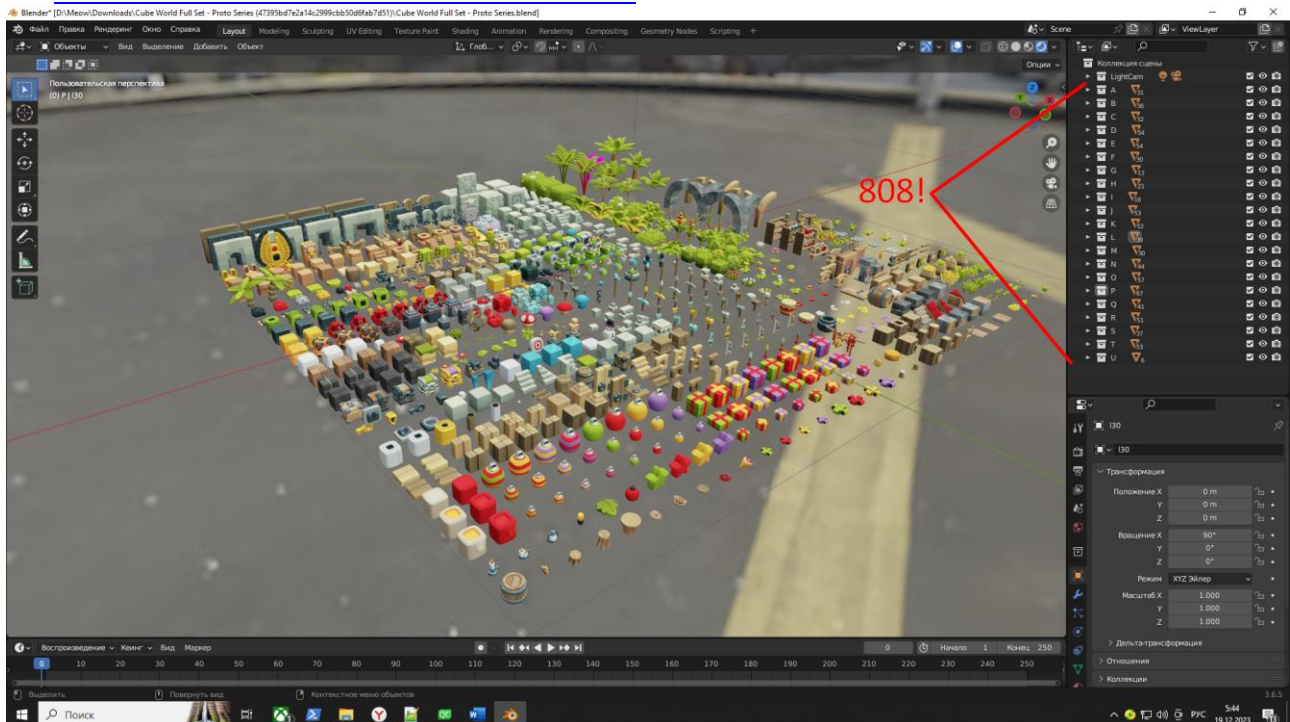


Рисунок 4. Вид моделек в блендере.

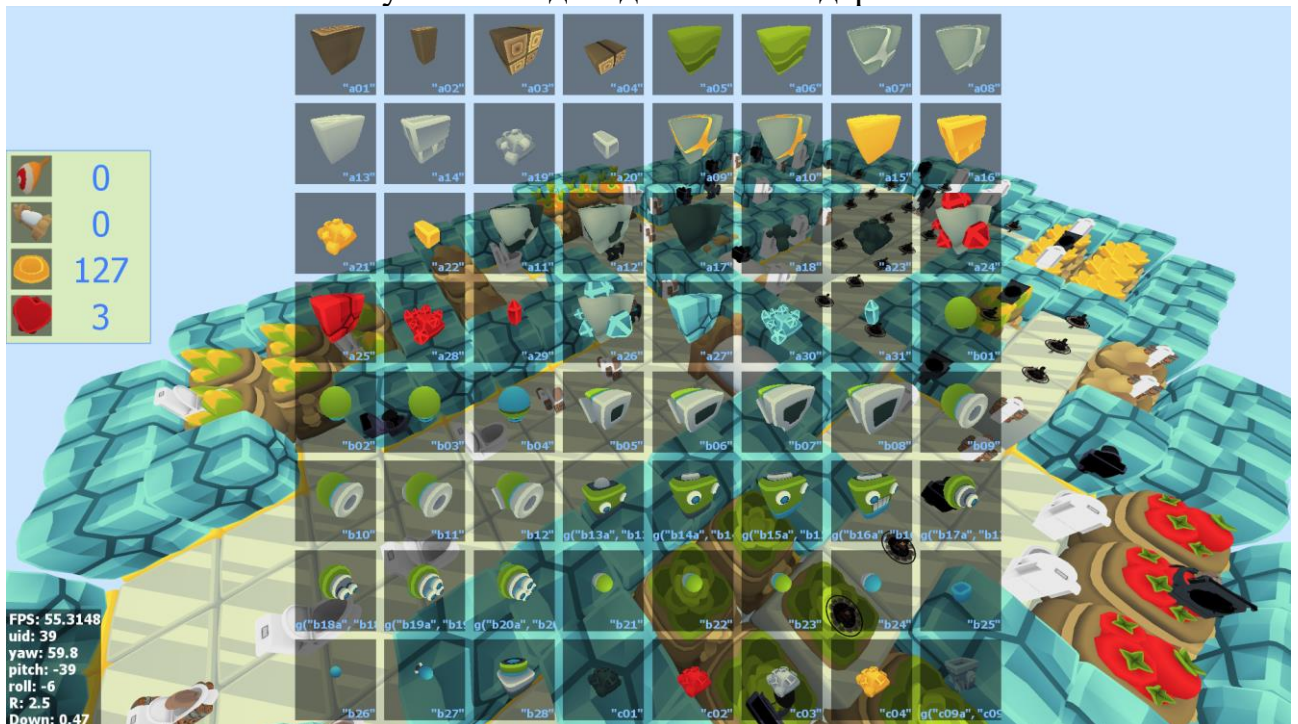


Рисунок 5. Вид первой страницы отображения моделек.

Могу подметить, что хоть в моём .asd-файле хранятся все модельки для строительства, все 25 питомцев и все 4 вида скибиди-туалетов. В моей программе загружается только часть из них в специальном индексе над моделями. Вот данный отображатель 8x8 показывает только то, что в индексе.

## 4. Прототип интерфейса

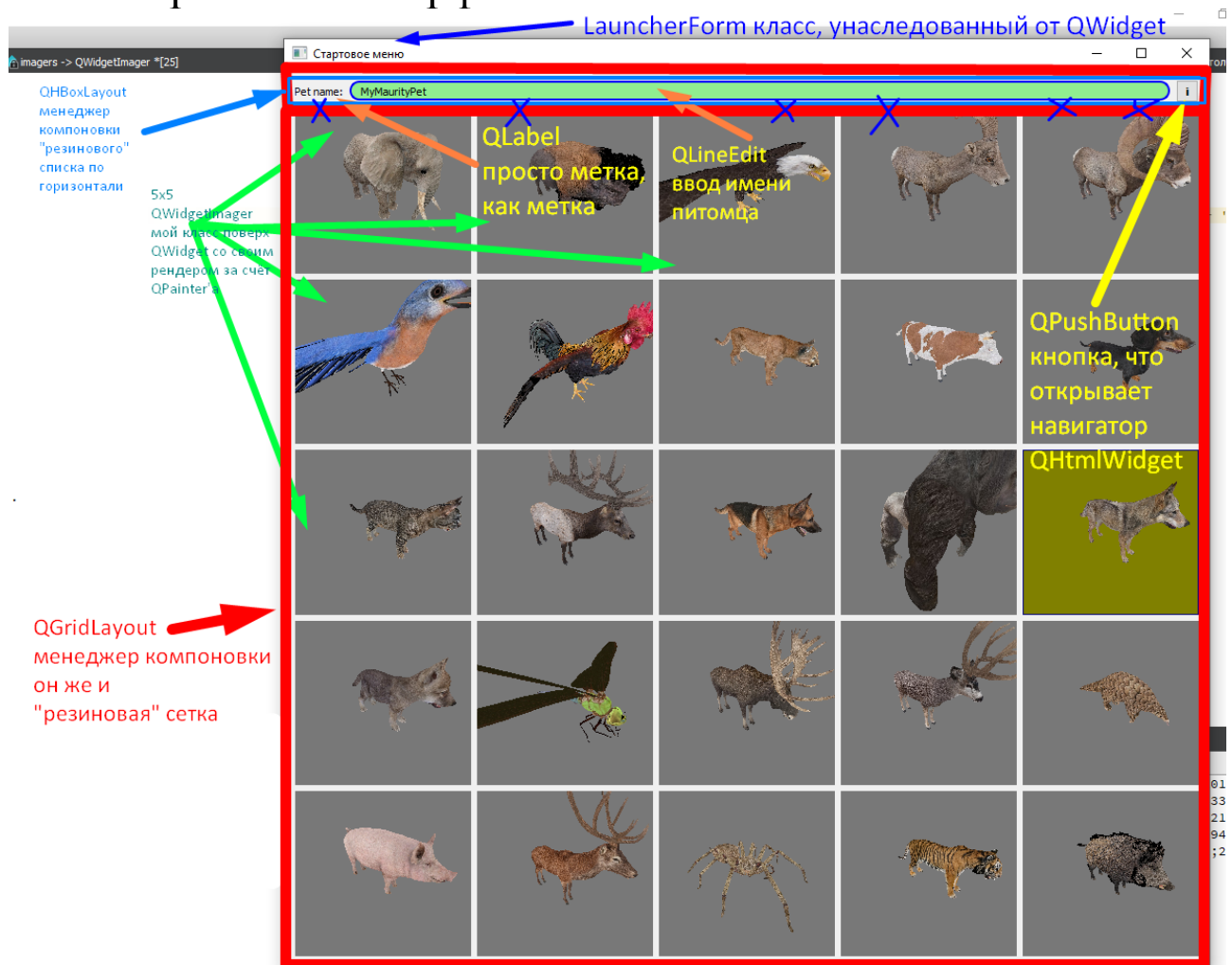


Рисунок 6. Вид стартового окна

Описание элементов интерфейса:

Кнопка 'i': открывает навигатор.

Поле ввода: позволяет ввести имя питомца. Каждое изменение сразу сохраняется.

25 кнопок QWidgetImager: отрисовывают двухмерную проекцию трёхмерной модели, а также по клику позволяют выбрать вид питомца. Каждый клик сразу же сохраняет выбор.



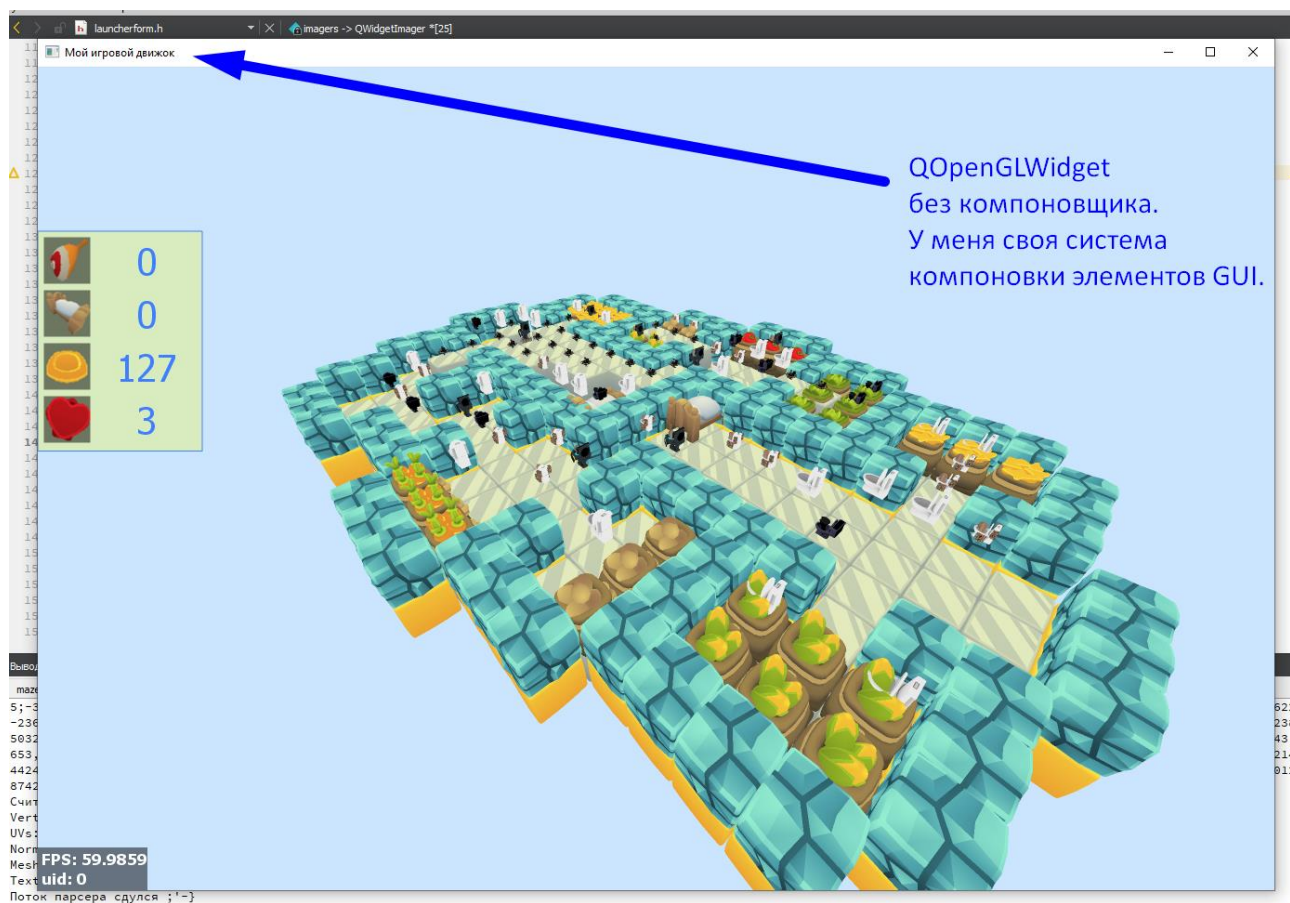


Рисунок 7. Вид главного окна.

Основное управление данного окна описано в навигаторе.

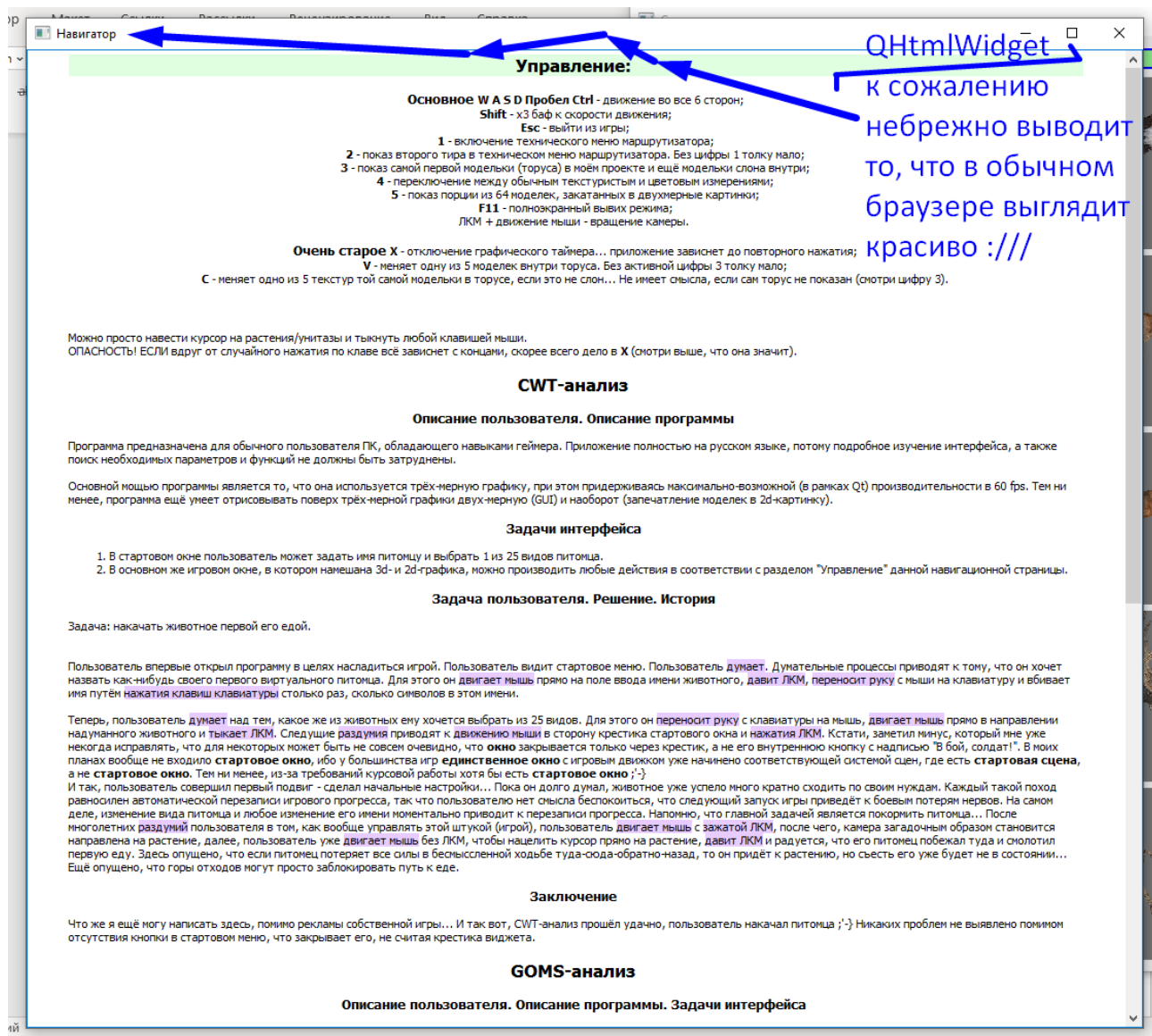


Рисунок 8. Навигатор.

Управление:

Основное

WASDПробелCtrl

Shift

Esc

1

2

3

4

5

F11

ЛКМ + движение мыши - вращение камеры.

Очень старое

X

V

C

Можно просто навести курсор на растения/унитазы и тыкнуть любой клавишей мыши.  
ОПАСНОСТЬ! ЕСЛИ вдруг от случайного нажатия по клавише всё зависнет с концами, скорее всего дело в X (смотри выше, что она значит).

CWT-анализ

Описание пользователя. Описание программы

Программа предназначена для обычного пользователя ПК, обладающего навыками геймера. Приложение полностью на русском языке, потому подробное изучение интерфейса, а также поиск необходимых параметров и функций не должны быть затруднены.  
Основной мощью программы является то, что она использует трёх-мерную графику, при этом придерживаясь максимально-возможной (в рамках Qt) производительности в 60 fps. Тем ни менее, программа ещё умеет отрисовывать поверх трёх-мерной графики двух-мерную (GUI) и наоборот (запечатление моделей в 2d-картинку).

Задачи интерфейса

1. В стартовом окне пользователь может задать имя питомцу и выбрать 1 из 25 видов питомца.

2. В основном же игровом окне, в котором намешана 3d- и 2d-графика, можно производить любые действия в соответствии с разделом "Управление" данной навигационной страницы.

h/basic.html

GOMS-анализ

Описание пользователя. Описание программы. Задачи интерфейса

Смотрите соответствующие пункты CWT-анализа. Разница только в том, что пользователю уже наиграл 10000 часов в данной игре.

Задача пользователя. Решение. История

Задача: питомец уже пережил миллион лет. Пользователь непрежно относился к уборке туалетов за питомцем. В итоге вся карта испещрена туалетами. Напомним, что если питомец ходит в один и тот же тайл 5 раз подряд, то ходяба через него блокируется. ЦЕЛЬ тажа - нужно покорить питомца, предварительно очистив карту от СОТНИ унитазов...

Данную ЦЕЛЬ я разобью на несколько ПОДЦЕЛЕЙ:

1. Запуск игры

2. Очистка СОТНИ унитазов. Допустим, что после каждой ДВАДЦАТКИ питомец устанёт и до кровати ещё по ПЯТЬ унитазов.

3. Ура, еда!

Хочу выделить такую СИСТЕМУ МЕТОДОВ:

1. Заккрытие стартового окна. Тыкаем ЛКМ по крестнику виджета. (МРВ)

2. Уборка унитаза. Двигаем камеру мышью с ЛКМ на цель, пока не увидим её, направляем курсор без ЛКМ и тыкаем ЛКМ. (МРВРВ)

3. Сон. Тежи действия, что и при методе Уборка унитаза. (МРВРВ)

4. Поедание. Тежи действия, что и при методе Уборка унитаза. (МРВРВ)

5. Движение камеры. Переносим руку с мыши на клавишу, давим ровно одну из кнопок WASDПробелCtrl, возвращаем конечность на мышь. Да! Нормальные игроки эти клавиши дают левой рукой, а мышь правой. Но! В рамках GOMS-анализа пользователь на столько опытный, что потерял левую руку из-за этой игры. Так что используем, что ещё пока есть :-} (МНКН)

Решение первой ПОДЦЕЛИ:

Здесь считается, что пользователь опытный, значит и питомец уже давно настроен. По этому достаточно метода Заккрытие стартового окна: (МРВ)

Решение второй ПОДЦЕЛИ:

А вот здесь уже нужно уточнить, что перед каждым выполнением блока из ПЯТИ методов Уборка унитаза, либо Сон, либо Поедание, стоит ставить метод Движение камеры, чтобы максимально подвести GOMS-анализ к правдоподобию: (МНКН + 5\*(МРВРВ)). Т.е. пользователю не получается всегда только повернуть камеру, чтобы увидеть свою цель-3d-объект. Вот где-то в среднем раз в ПЯТЬ вращений головой идёт ОДНО телодвижение даже для опытного пользователя. В данной подцеле у нас всего СОТНЯ унитазов, а это значит что таких пАтёрков будет 20. Помимо этого, как раз придётся произвести одну пАтёрку движений к кровати. В сумме будет 21 пАтёрка : (21\*(МНКН + 5\*(МРВРВ)))

Решение финальной ПОДЦЕЛИ:

По сути это первое и последнее использование метода Поедание в неполной пАтёрке, по этому, разумею всего, будет склеять этот метод с методом Движение камеры: (МНКН + МРВРВ)

В итоге мы получаем такую формулу:  
$$(МРВ) + (21*(МНКН + 5*(МРВРВ))) + ((МНКН) + (МРВРВ)) =$$
$$= (МРВ) + ((21-1)*(МНКН) + (21*5-1)*(МРВРВ)) =$$
$$= (МРВ) + (22*(МНКН) + 106*(МРВРВ)) =$$
$$= 129*М + 213*Р + 213*В + 44*Н + 22*К =$$
$$= 129*1.38 - 213*1.1 + 213*0.2 + 44*0.4 + 22*0.2 =$$
$$= 473.05 \text{ секунд, т.е. } 7:53.05 \text{ в формате минуты:секунды.сантисекунды соответственно. Ура!}$$

Заключение

Довольно правдоподобное получилось значение. По сути можно округлить до ВОСЬМИ минут. По сути игрок разгрёб 100 туалетов, 5 раз поспал и 1 раз съел финальное растение. Осталось во всех училищах геймеров ввести норматив, в котором ВОСЕМЬ минут будут на оценку '3', если такие вообще когда-нибудь появятся, не считая летсплеи на ютубе. Оценку '4' и '5' можно будет просчитать по этой же формуле, слегка уменьшив все эти буквы МРВНК. А что, не плохое применение данного GOMS-анализа :-}

Спасибо за внимание!!!

Рисунок 9 и 10. Оригинальный вид навигатора.

## 5. Реализация

Требование	Описание
реализация основных задач пользователя по выбранной теме	Питомец есть, ходить ногами умеет, выдавать отходы жизнедеятельности умеет, питаться растениями умеет. Слушается пользователя, а если нет задачи, хаотично ходит по игровой карте. Имеет свой уровень усталости и голода.
работа с файлами	Очень обильная работа с файлами, т.к. в ресурсный файл внедрён .asd-файл набора моделей; старые .obj-файлы и текстуры даже не от них; вершинный и фрагментный шейдер; html-файл навигатора и css-файл стилей всего проекта.
использование SQLite	Разобрано создание соединения, создание таблицы (если её ещё нет), запись данных в таблицу и считывание данных и из таблицы. Всего-лишь разобран этот аспект и проведены тесты по времени посредством QTime.
проверки на ввод некорректных значений в программе	Разработка так велась, что ни одно тестирование не показало в процессе разработки, что есть возможность сломать как-то проект. Любые краши с новых изменений были моментально выявлены и устранены.
CWT, GOMS	Полностью реализованы! Внедрены в навигатор после раздела с Управлением.
использование таймера	С первого же вздоха проекта таймер был использован для вызова события repaint, без которого QOpenGLWidget попусту не будет ничего рисовать.
информационная наполненность	Сам факт 3d-мира и множество профессиональных 3d-моделей будоражит многие фантазии детей, что любят играть в подобные 3d-игры.
сохранение результатов предыдущей работы пользователя	Сохраняется в принципе весь прогресс: положение зверька, голод, усталость, богатство (money <=> score), количество использований кровати для восстановления сил, сколько и какое растение растёт (но ещё не выросло, ибо выросшие растения не сохраняются, т.к. это значение по умолчанию), где и в каком количестве питомец сходил по своим нуждам в скибиди-туалет.
использование менеджеров компоновки на всех формах	Обильно использовался на стартовом виджете. Вообще не использовал на QOpenGLWidget по тому, что он не рассчитан вообще для какой-либо компоновки, но позволяет писать <b>свою собственную</b> компоновку на уровне 2d-графики. Если бы не этот пункт с требованиями, стартового меню вообще бы не было, но это не факт.
реализация навигатора справки	Всё равно планировал где-то оставить информацию об управлении игрой. Ещё использовал, как хранилище для CWT- и GOMS-анализов.

использование отдельного файла css для всего проекта	На самом деле это из разряда работы с файлами. Отличается это требование от работы с файлами всего лишь одной строчкой. Так вышло, что у меня изначально объект QApplication был протолкнут в стартовое меню для передачи полного спектра событий с интерфейсов управления для одной функции-фильтра.
--	---

## Функционал

Для подробностей, обратитесь к разделу “Управление” в навигаторе.

Основная задача данного приложения – поддерживать игроку жизнь в питомце. Но что же такое жизнь питомца? В рамках моей программы, это всего лишь:

Голод – то, сколько осталось единиц сытости до полного истощения питомца

Усталость – то, сколько действий питомец ещё может совершить перед тем, как захочет спать. Суть в том, что он не может выполнять действия при усталости=0, но и не может спать в обратном случае.

Богатство (деньги) – по сути очки в этой игре. Деньги не на чего тратить, зато они отображают объём проделанной работы для поддержания жизни питомца.

Кровать (сон) – то, сколько раз питомец поспал за его жизнь.

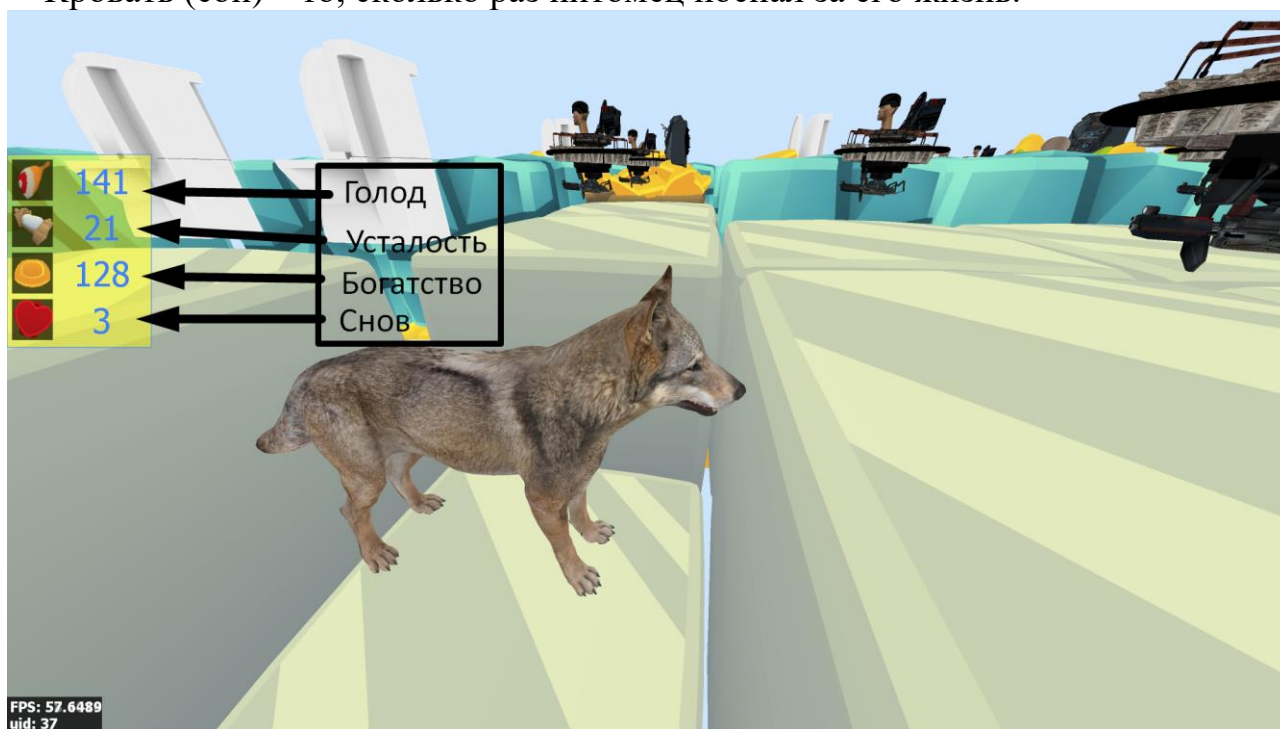


Рисунок 11. Основные характеристики жизнедеятельности питомца.



Для того, чтобы убрать отходы, нужно на них навести и кликнуть любой кнопкой мыши:

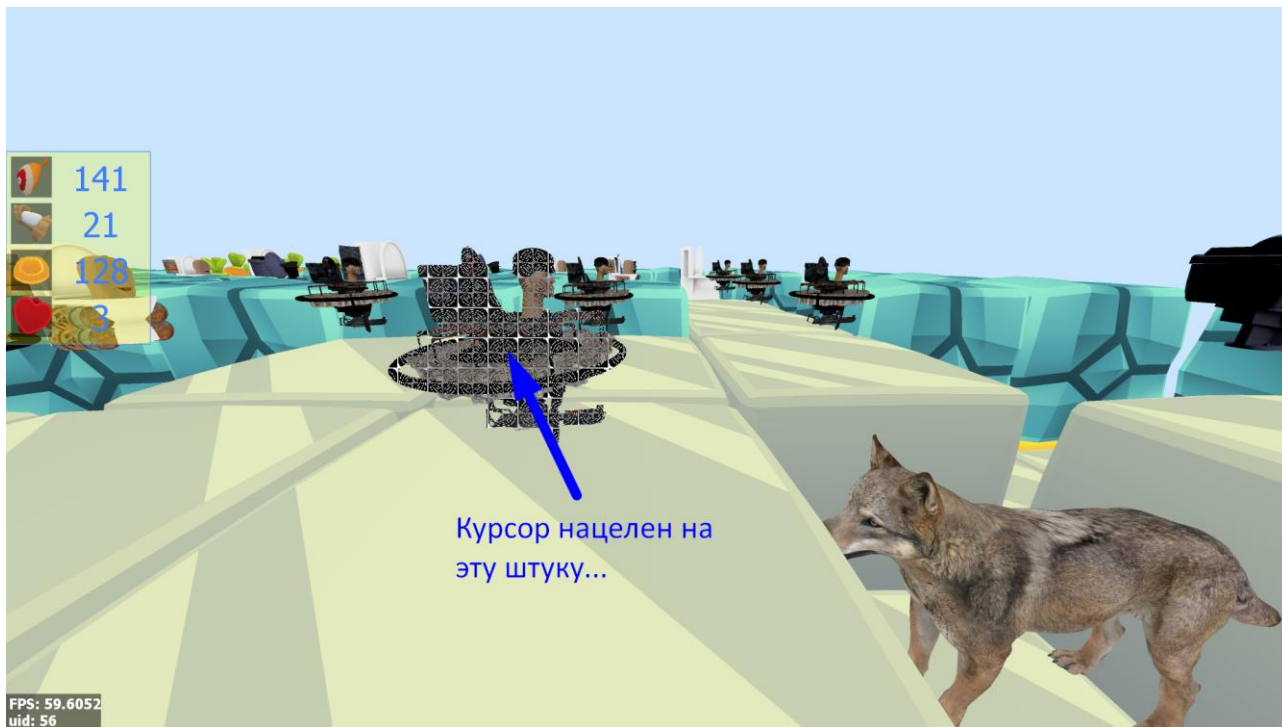


Рисунок 12. Нацеленный курсор на цель в виде отходов питомца.

Здесь отчётливо видно, что отходы жизнедеятельности блокируют дорогу питомца. Единственным исключением является попытка питомца убрать их для того, чтобы попасть к кровати, если силы иссякнут.

Если хотя бы есть 2 единицы усталости, то отнимаются 2 единицы усталости и прибавляется одна единица богатства. Если нет 2 единиц усталости, то удаление отходов невозможно.

В случае с едой всё почти тоже самое:

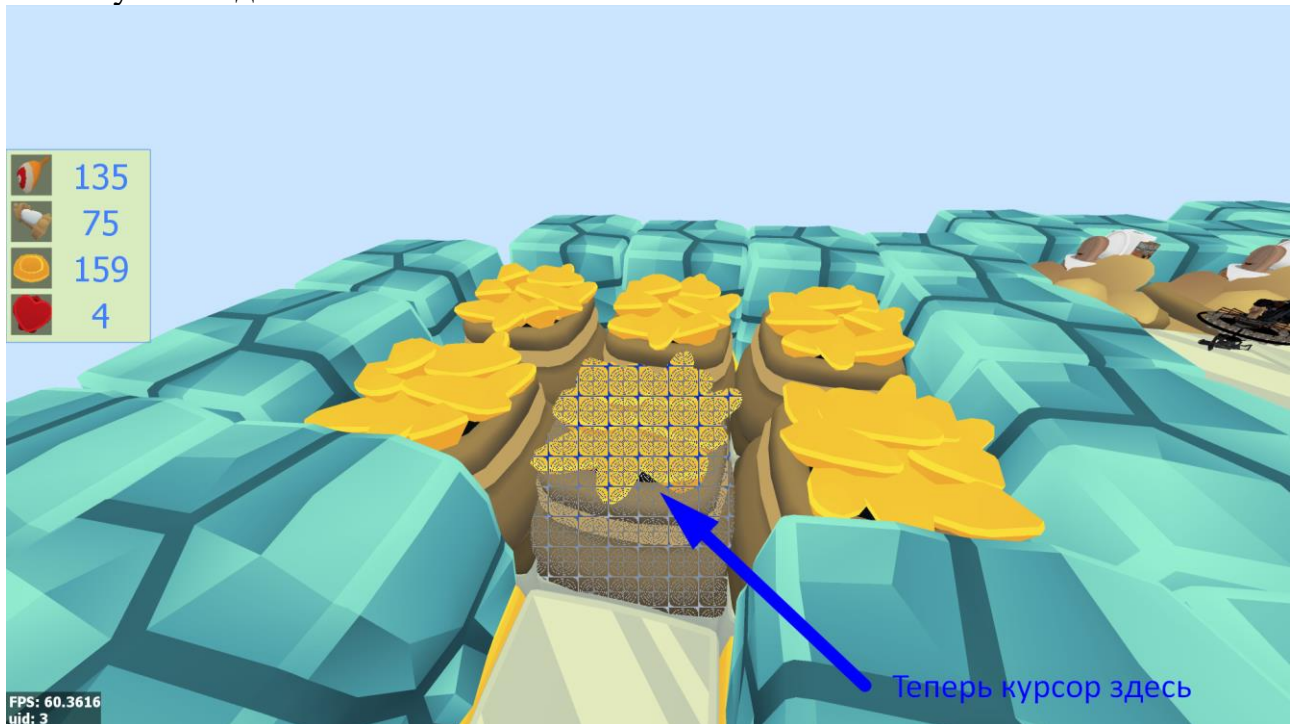


Рисунок 13. Нацеленный курсор на еду.

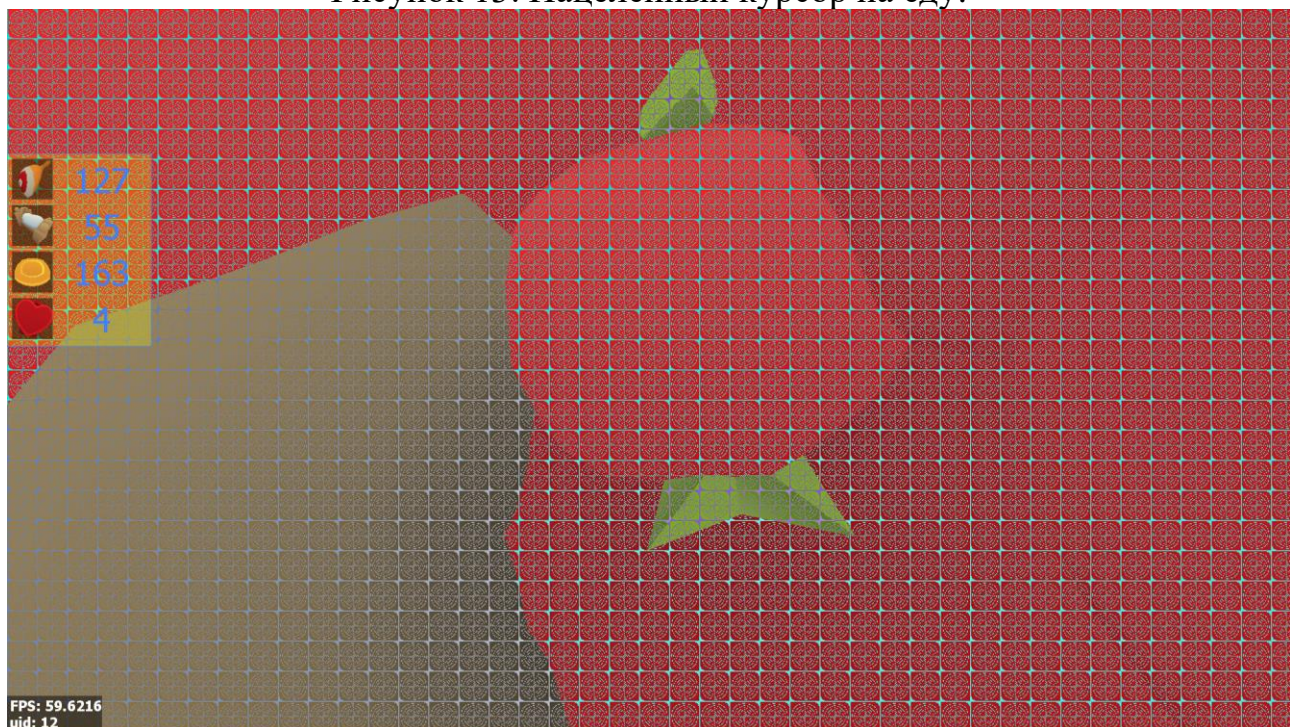


Рисунок 14. Вид шейдера – выделителя цели вблизи.

Если хотя бы есть 3 единицы усталости, то отнимаются 3 единицы усталости, добавляются 3 единицы еды и прибавляется одна единица богатства. Если нет 3 единиц усталости, то поедание еды невозможно.



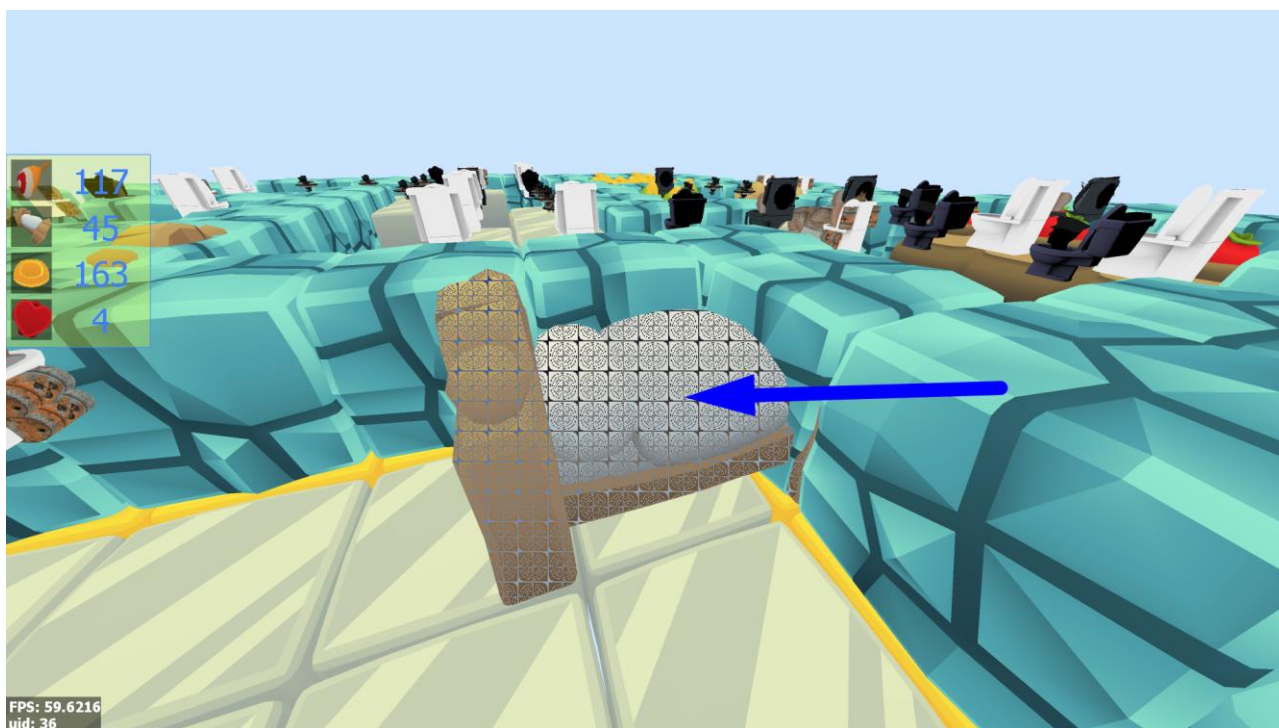


Рисунок 15. Цель – кровать.

Если хотя бы есть 1 единица усталости, спать нельзя. Если единиц усталости 0, сон в кровати приводит к восстановлению усталости в первоначальное значение = 99, прибавляется аж целые 10 единиц богатства и 1 единица сна – возраст питомца.

Ещё нужно понимать, что питомец каждые 3 секунды ходит в туалет только в том случае, если у него нет цели (что работает только при усталости не меньше 50 для стабильности). Нельзя сходить (в плане туалета) в ячейку, в которую нельзя сходить (в плане перемещения) – всё логично! ;'-} Это же и касается, если питомец просто застрял из-за того, что его окружили стены и/или туалеты последнего 5-уровня, т.е. не может идти, т.е. не в анимации.

Уже давно описано, из каких окон состоит приложение – из стартового для выбора имени и вида питомца, и для вызова навигатора, и из главного окна, где и происходит вся ранее описанная игровая движуха.

По сути, на этом кончается функционал Тамагочи питомцев и **только начинается** функционал игрового движка!!!

Вот что произойдёт при нажатии на цифру ‘1’ основной или боковой клавиатуры:



Рисунок 16. Техническое меню первого уровня маршрутизатора питомца.

А вот к чему приводит нажатие цифры ‘2’. Не имеет эффекта без цифры ‘1’:



Рисунок 17. Техническое меню второго уровня маршрутизатора питомца.

Не трудно догадаться, что для первого уровня используется обычный BFS, он же поиск в ширину, а для второго уровня уже произвольная трассировка линий.

Т.е. для того, чтобы проверить возможность пройти питомцу по диагонали между двумя ячейками, нужно проверить, что чёрные линии не касаются стен (ячейки вообще не рендерятся), либо красных ячеек, т.е. там навалена куча отходов 5-ого уровня.

Повторное нажатие цифр '1' и '2' обращают их эффекты вспять.

Нажатие на цифру '3' приводит к этому:

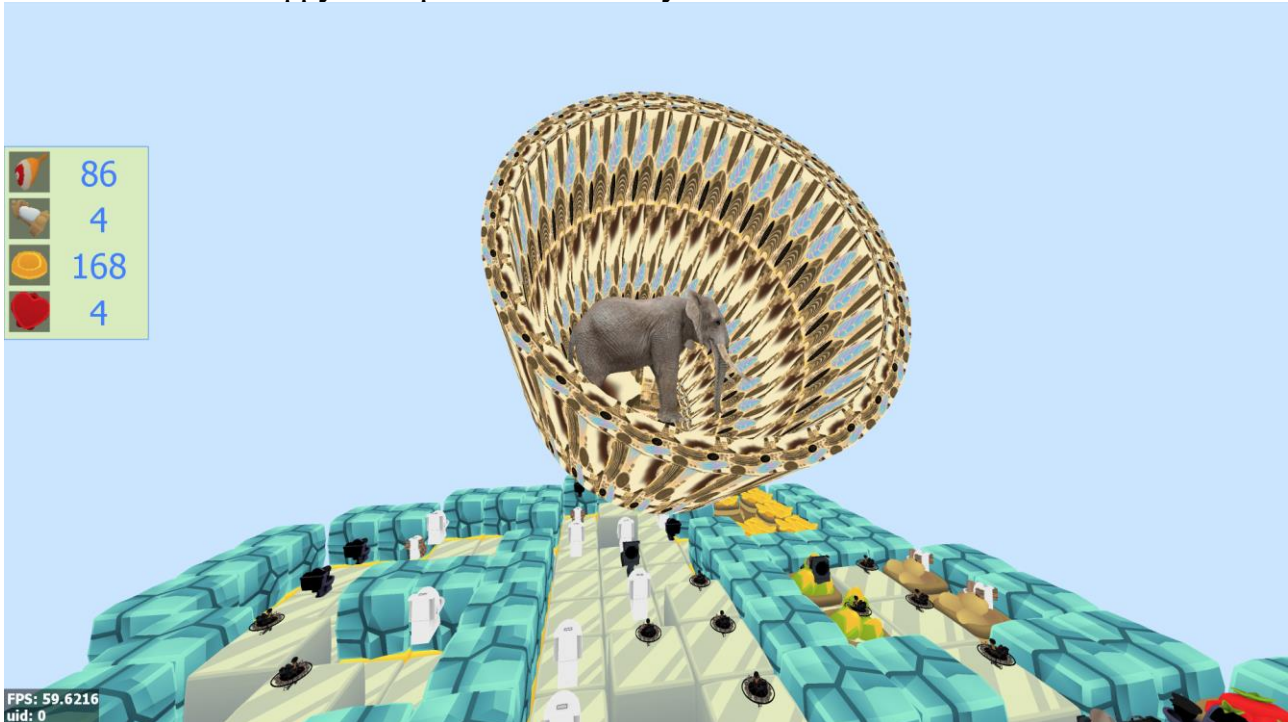






Рисунок 18. Торус со слоном внутри.

После этого имеет смысл давить кнопки 'C' и 'V'. Не буду на них за цикливаясь, смотрите Навигатор.







Эффект от цифры '4' самый потрясающий:

	73
	94
	178
	5



FPS: 59.6471  
uid: 39

	58
	79
	178
	5



FPS: 60.1094  
uid: 0

Рисунок 19 и 20. Альтернативное-цветовое измерение игры.

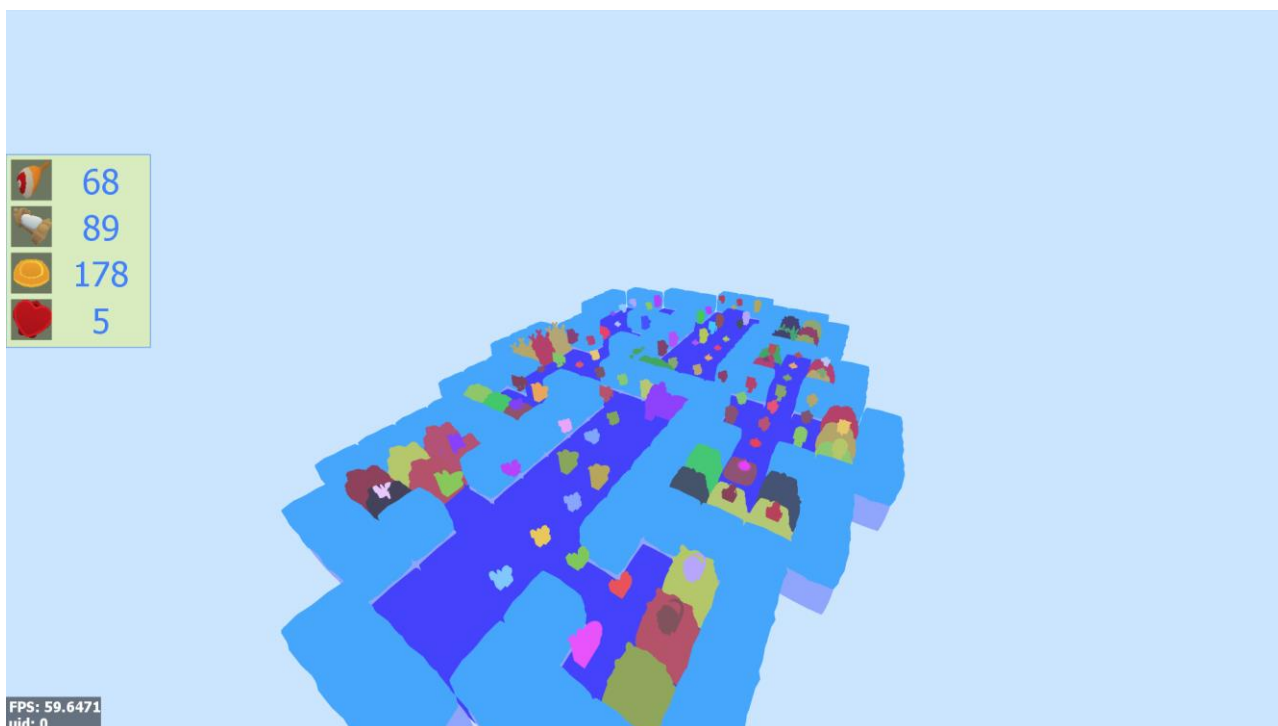


Рисунок 21. Альтернативное-цветовое измерение игры.

На рисунках 19-21 продемонстрировано альтернативное измерение в игре. Это цветовое измерение. Суть его заключается в том, что цвет, под которой находятся курсор, можно конвертировать в идентификатор модели/группы моделей. Это фундаментальная основа для определения, куда же направлен курсор или куда пользователь нажал. Разумеется, у меня реализована обширная система как отрисовки самих моделек, так и система различных событий над моделью:

- курсор зависает над моделью,
- курсор появился над моделью,
- курсор покинул модель,
- падение какой-то из кнопок мыши над модель,
- движение мыши с зажатой какой-то кнопкой мыши,
- движение курсора над моделью без зажатых кнопок,
- отпускание какой-то кнопки мыши после движения,
- просто клик по модели какой-то кнопкой мыши.

Стоит отметить, что зависание курсора над моделькой означает, что курсор указывает на модельку, а само событие срабатывает перед каждой отрисовкой сцены, т.е., обычно, 60 раз в секунду. Точное значение всегда пишется снизу-слева после надписи FPS, т.е. Frames Per Second (кадров в секунду). Появление и покидание модельки курсором – одноразовая акция, которая срабатывает только при изменении текущего UID, т.е. уникального идентификатора модельки, на который указывает курсор. Жаль, что не все игры могут похвастаться данной технологией, зато готовы подавать кривые хитбоксы.

Интересный факт, что от запуска к запуску приложения, одни и те же модельки имеют разные UID. Это связано с тем, что UID раздаются только в порядке необходимости и используются только для конвертирования UID модельки в сам объект модельки для вызова тех или иных её событий, если они вообще заданы, иначе вызов просто улетит в молоко, как, к примеру, клик кнопки мыши по блоку, не имеющим какой-либо цели (стены, пол). Очень сильно влияет на UID само содержимое файла сохранения прогресса.

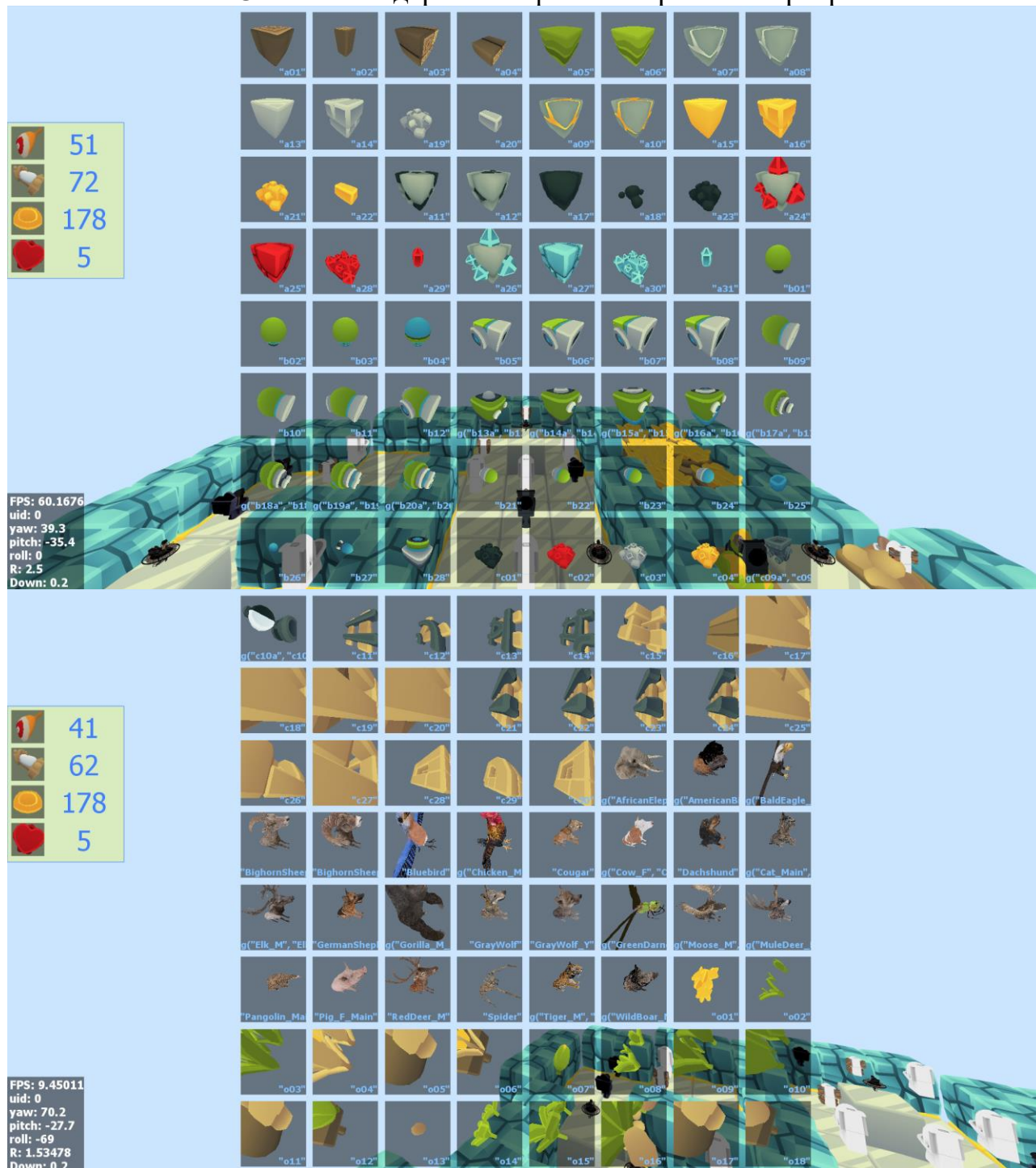


Рисунок 22 и 23. Первая и вторая страница последствий цифры '5'.



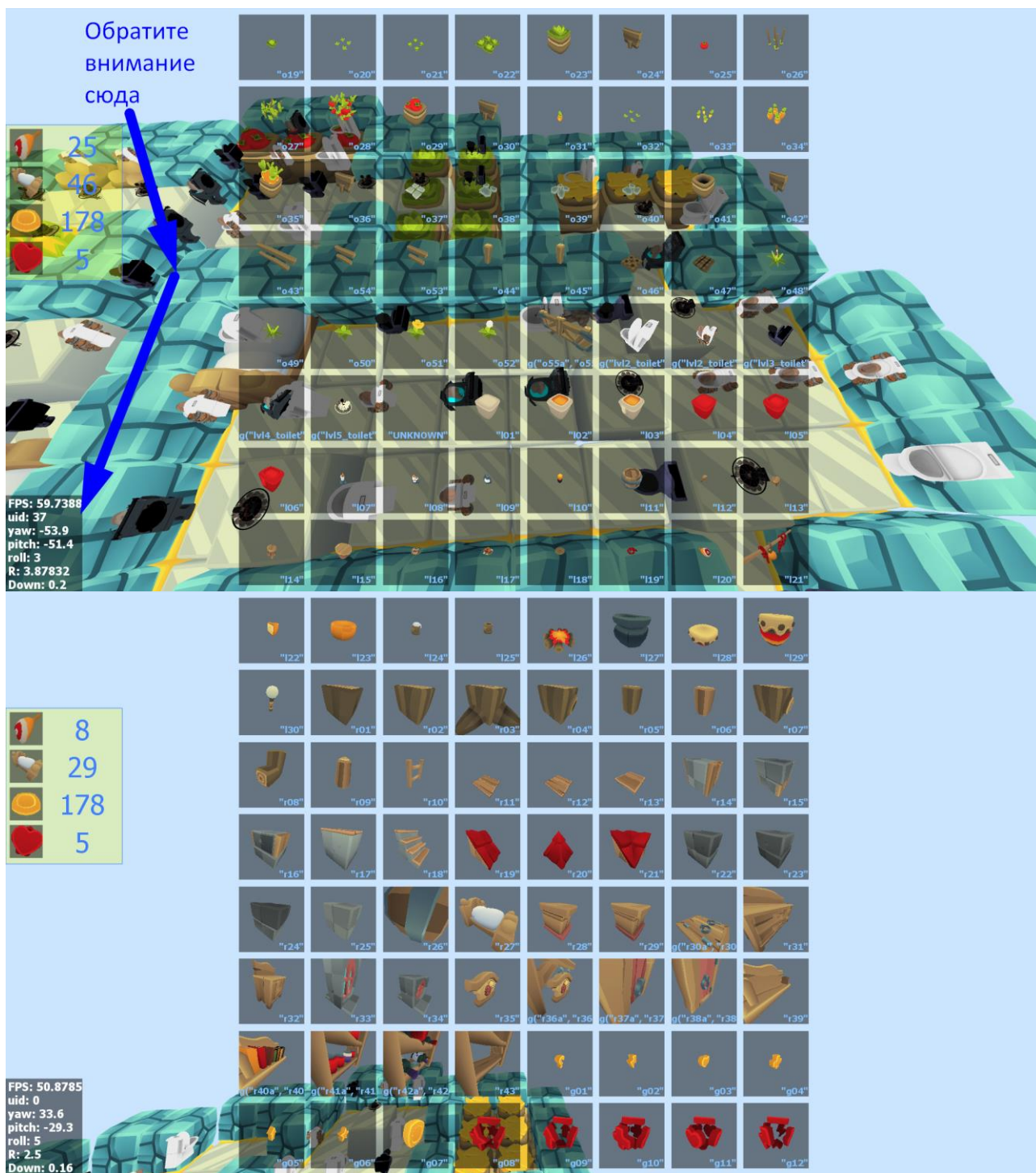


Рисунок 24 и 25. Третья и четвёртая страница последствий цифры '5'.

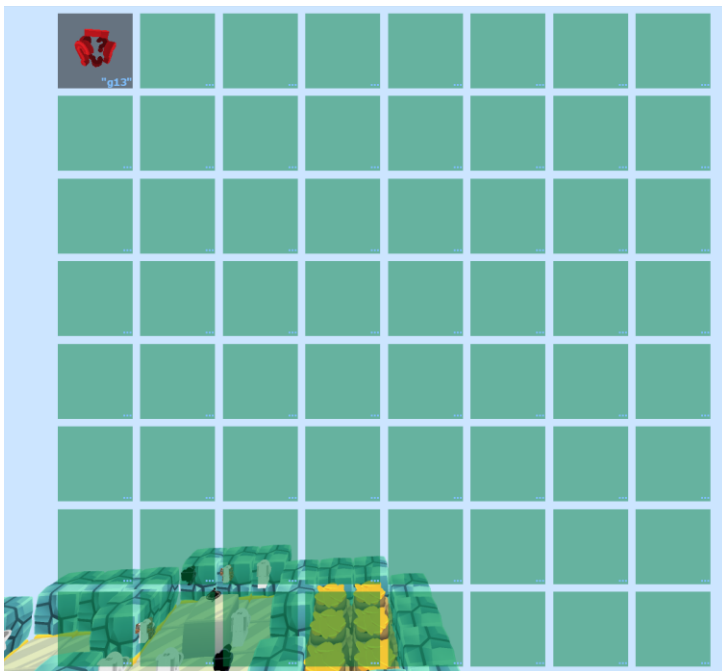


Рисунок 26. Последняя и забавная страница последствий цифры '5'.

Т.е. на страницу вмещается  $8 \times 8 = 64$  модельки. Всего моделек в индексе 257. Случайно и совершенно случайно так вышло, что на последней страницу только одна моделька. И всё это из-за одной модельки, что как таковой моделькой и не является, находится она на ID 170 и просто олицетворяет пустоту... Не совсем уже помню, для чего... Вроде бы я это сделал для union-группы скибиди-туалетов и нулевой уровень взял за пустую модельку, но сейчас в этом смысла мало, ибо сами туалеты попусту удаляются на уровне карты после их очистки питомцем.

Бирюзовая клетка означает, что был произведён выход за пределы индекса моделей. Напоминаю, что модели (808 строительных, 25 питомца и 4 скибиди-туалета) и индекс моделей (в нём зарегистрировано, т.е. индексировано всего 257 моделей) – это две разные вещи. В индекс модели попадают только по делу, а не просто по тому, что они были в наборе и просто попали в мой .asd-файл с перечислением моделек. ;' - } На самом деле я в индексы добавляю строительные модели группами, так что даже в индексе часть моделек попусту не используется.

Дополнительное управление простое – движение мыши + ЛКМ меняет yaw и pitch. Клавиши 'A' и 'D' клавиши меняют roll. Колесо мыши отдаляет и приближает (параметр "R", т.е. радиус). Клавиши 'W' и 'D' меняют параметр "Down", т.е. смещение модели относительно вертикали.

На этом и оканчивается весь чудесный технический функционал данной курсовой работы.

Приложение состоит из окон: основное окно и стартовое окно. Они уже описаны в разделе “ПРОТОТИП ИНТЕРФЕЙСА” как и их скриншоты с нарисованным поверх них прототипом.

Прежде чем перейти к структуре базы данных игрового прогресса, взгляните на сие записки охотника:

```

21
22 typedef unsigned char byte;
23 typedef unsigned short ushort;
24 typedef long long int long8;
25
26 #define UNDEF_POS 0x7fffffffffffffff
27 #define X_Y_TO_POS(x, y) (long8(x) + 0x80000000 + long8(y) * 0x100000000)
28 #define PRINT_POS(pos) (qDebug("pos: %d %d", int((pos & 0xffffffff) - 0x80000000), int(pos >> 32)));
29 #define POS_TO_X_Y(x, y, pos) do { x = int((pos & 0xffffffff) - 0x80000000); y = int(pos >> 32); } while(false);
30
31 static int props[4] = { 99, 99, 0, 0 };
32

```

Рисунок 27. Записки охотника.

Нам важным только макросы для понимания того, как X и Y превращаются просто в число типа long8... Из-за этих макросов пришлось выработать самому пару техник безопасности составления макросов, иначе будет получаться не то, что хотелось бы. Эти техники безопасности я не знал, но пришлось встретиться с ними, самостоятельно переработать в голове, почему не работает та или иная ситуация и понять, как фиксировать дальше. В целом мне помогло простое понимание сути макросов на уровне препроцессора Си-языка.

Если бы требовалось в long8 затолкать координаты X, Y и Z. Тогда я бы выделил на это 21, 22 и 21 битов соответственно, вместо 32 и 32 битов для X и Y соответственно. На Y (координата высоты) приходился бы лишний бит неспроста, а их банальной логики, что в кубических играх игроки падают (или зарываются в шахтах) чаще, чем двигаться по горизонтали, ибо так веселее.

Структура базы данных игрового прогресса такова:

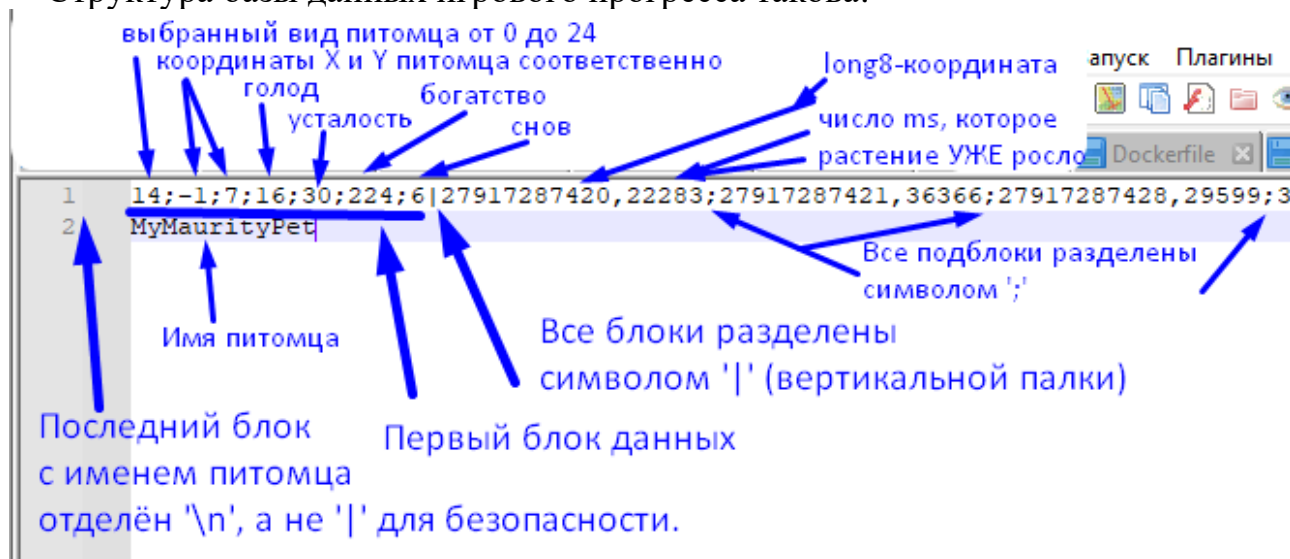


Рисунок 28. Краткое пособие об структуре прогресса игры.

Довольно-таки простая структурка, в которой **все данные** делятся на **блоки** посредством символа ‘|’ (исключением последний блок, перед которым ‘\n’ вместо ‘|’). Все **блоки** делятся на **подблоки** посредством символа ‘;’.

Все числа (подблоки) самого первого блока я уже описал на скриншоте.

Все подблоки второго и третьего блока состоят из пар чисел, разделённых запятой. Левое число всегда означает координату в формате long8 (смотрите те самые макросы на рисунке 27).

Второй подблок олицетворяет именно съеденные растения, а правое число в паре означает, сколько миллисекунд (ms) после этого прошло, если проходит 5 минут, растение полностью вырастает и больше не нуждается в записи в файл.

Третий блок олицетворяет отходы жизнедеятельности питомца. Левое число пары снова же означает координату в формате long8, а правое – уровень отходов от 1 до 5. Отходы с уровнем 0 не хранятся, ибо нет смысла, т.к. по умолчанию их вообще нет, т.е. при первом запуске игры.

CWT- и GOMS-анализы смотрите в Навигаторе (рисунки 8-10)

## 6. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Описание оригинальных Тамагочи питомцев - <https://hipilion.livejournal.com/31611.html>
2. Крутая система уроков по OpenGL - <https://triplepointfive.github.io/ogltutor/>
3. Ещё один сайт с уроками по OpenGL - <https://pmg.org.ru/nehe/> здесь я чаще всего провожу время.
4. Малоизвестный русскоязычный YouTuber, скорее всего преподаватель предмета, посвящённого OpenGL - [https://www.youtube.com/watch?v=H3jt5m6U\\_Jk&list=PL-hrQhpTB95LKMbttX47vCsNeGbJQVz1-&index=17](https://www.youtube.com/watch?v=H3jt5m6U_Jk&list=PL-hrQhpTB95LKMbttX47vCsNeGbJQVz1-&index=17) это как раз тот самый ролик про идею с цветовым измерением. Увы, автор данного ролика не совсем разбирается в самом Qt, только в самом OpenGL и C++, так что более простой способ вывести цветовое измерение он не догадался сделать, как я в данном курсовом проекте.
5. Всё. Мне много и не надо для подвигов... Не считая колоссальное количество форумов, что я перелопатил для исправления тех или иных мелочей в Qt или узнавания чего-то новенького по Qt.

## 7. ПРИЛОЖЕНИЕ

В следствии того, что из-за обильного количества кода (свыше 4000 строчек кода в сумме) приложение сильно раздует, даже если заикливаться на самом важном, преподаватель практики разрешил внести сюда только ссылки на репозитории:

Основа: <https://github.com/VectorASD/VPICHMV>

Зеркало: <https://git.csc.sibsutis.ru/ip111s19/VPICHMV>

Зеркало добавлено только на тот случай, если основа (github.com) выйдет из строя, возможно навсегда. Да и мне так спокойнее просто.