

Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра вычислительных методов и программирования

Д. П. Кукин, Т. А. Рак, О. О. Шатилова

ИНДУСТРИЯ КОМПЬЮТЕРНЫХ ИГР. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

*Рекомендовано УМО по образованию в области информатики
и радиоэлектроники в качестве пособия для направления специальности
1-40 05 01-12 «Информационные системы и технологии
(в игровой индустрии)»*

Минск БГУИР 2019

УДК 004.928(076.5)
ББК 32.972.13я73
К89

Рецензенты:

кафедра программного обеспечения вычислительной
техники и автоматизированных систем
Белорусского национального технического университета
(протокол №6 от 17.01.2017);

доцент кафедры экономической информатики учреждения образования
«Белорусский государственный экономический университет»
кандидат экономических наук, доцент О. А. Синявская;

технический писатель ООО «Мелсофт» М. В. Кравченко

Кукин, Д. П.
К89 Индустрия компьютерных игр. Лабораторный практикум : пособие /
Д. П. Кукин, Т. А. Рак, О. О. Шатилова. – Минск : БГУИР, 2019. – 60 с. : ил.
ISBN 978-985-543-467-3.

Содержит теоретический обзор курса, а также индивидуальные задания по
лабораторному практикуму по курсу «Индустрия компьютерных игр».

УДК 004.928(076.5)
ББК 32.972.13я73

ISBN 978-985-543-467-3

© Кукин Д. П., Рак Т. А., Шатилова О. О., 2019
© УО «Белорусский государственный
университет информатики
и радиоэлектроники», 2019

Лабораторная работа №1

Часть 1

ЗНАКОМСТВО С XMIND. СОСТАВЛЕНИЕ ИНТЕЛЛЕКТ-КАРТ С ОПИСАНИЕМ КОМПЬЮТЕРНОЙ ИГРЫ

Mindmap – это диаграмма связей или «Интеллект-карта» (ИК).

ИК – визуальное представление процесса или структуры в виде дерева ассоциаций, расходящихся от центрального понятия.

Правильно составленная карта позволяет за несколько минут воспроизвести большой поток информации. А правильно составляемая карта позволяет вам сразу, во время восприятия большого количества информации, быстро сориентироваться и найти главное для вас. Правила составления разработаны психологом Тони Бьюзенем и изложены в его книге «Супермышление».

Что такое mindmap и как их рисовать

Mindmap, или Карты Ума, Карты Сознания (не путать с когнитивными картами) – это кустообразные картинки, изобретенные Тони Бьюзенем как способ представления и связывания мыслей. Рисуются они очень просто:

1. Берем лист бумаги и в центре картинкой или одним-двумя словами обозначаем основное понятие или концепцию либо анализируемую проблему.
2. Обводим это понятие в кружок – из него все будет «произрастать».
3. Рисуем ветки (рис. 1.1) – основные связанные понятия, свойства и т. п. Подписываем их одним-двумя словами каждую (получается далеко не всегда, тем более русские слова достаточно длинные).

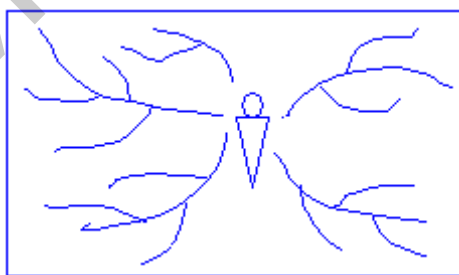


Рис. 1.1. Ветки mindmap

4. Далее – больше. Каждая ветка делится на несколько более тонких веточек – развитие этих понятий, детализация свойств, направлений работы.

5. Лист постепенно заполняется, и проблема из страшной-непонятной становится вполне решаемой – приколоты тоненькими веточками как булавками к листу бумаги.

6. Берем цветные ручки и внимательно смотрим на получившуюся картинку. И видим, что приколата проблема к листу кривовато, надо бы исправить. Обычно рекомендуется рисовать карандашами, чтобы иметь возможность стереть и исправить. Лучше не стирать, пусть на бумаге останется история мысли, а цветными ручками обвести некоторые веточки. Каждый цвет – своя связь. При некоторой тренировке очень легко на одной картинке видеть две схемы – на одной связи обозначены иерархией веток, на другой – одинаковой раскраской.

7. Теперь, если картинка получилась красивой и достаточно запутанной, повесьте ее на стену. И проблема вся как на ладони, и гости могут задавать вопросы.

8. Если карта изображает некий проект (план действий), то цветным карандашом можно обводить выполненные задачи. Для этого необходимо выбирать неяркие цвета, чтобы прежде всего видны были задачи к исполнению. Таким образом, наблюдательные посетители могут тренировать свою память – что же изменилось с последнего посещения. Это очень стимулирует.

Чем хороши mindmap и их свойства

1. *Наглядность.* Конечно, mindmap – всего лишь древовидная структура и этим эквивалентна многоуровневому линейному списку (пример – оглавление). Но есть и серьезные отличия:

а) Mindmap позволяет окинуть взглядом все сразу – все дерево. Линейный список не дает такой возможности.

б) Линейный список психологически не завершен, он может быть продолжен. Ничто не мешает дописать что-то в конец (да и в середину тоже – естественных рамок нет). Mindmap как рисунок ограничен размерами листа. Приходится выбирать существенное и не делить на большое количество частей.

2. *Эстетичность.* Mindmap – это рисунок, который имеет свою эстетику. Вернее – должен иметь, т. е. если mindmap как рисунок неприятен, то бесполезен и даже вреден. Создатель техники mindmap Тони Бьюзен специально подчеркивал, что при создании таких схем существенно включается в работу правое полушарие мозга, отвечающее за эстетику и холистический подход. То есть, рисуя mindmap по какой-то проблеме, мы обдумываем ее частью мозга, которая, как правило, не участвует в процессах принятия решения. Всегда полезно рассмотреть проблему с разных сторон.

3. *Формы, или 8×4.* Рисование mindmap происходит одновременно с осмыслением проблемы, оно и есть это осмысление. Мы помним, что рисунок должен быть эстетичным. Вместе с тем осмысление есть процесс, т. е. mindmap растет, как дерево или куст – без жесткого плана (это не архитектура), просто заполняя пространство. Такое эволюционное развитие рисунка существенно отличает его от чертежей. В качестве примера можно привести следующие типовые формы mindmap, каждая из которых легко переходит в следующую:

3.1. Андреевский флаг – 4 луча (рис. 1.2).

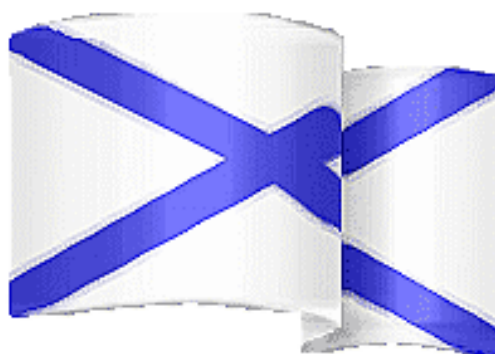


Рис. 1.2. Андреевский флаг

Как правило, у mindmap более 3 лучей. Лучшая стартовая форма – диагональный крест (Андреевский флаг): достаточно просто и есть место для новых веток, которые могут появиться при дальнейшем рассмотрении проблемы.

3.2. Звезда – 5 лучей (рис. 1.3).



Рис. 1.3. Звезда

У Андреевского флага появляется верхний луч – места много, эта ветвь может быть столь же расширяемой, что и предыдущие.

3.3. Снежинка – 6 лучей (рис. 1.4).



Рис. 1.4. Снежинка

У Звезды вырастает нижний луч. Если построение карты начиналось с Андреевского флага, то места хватает.

3.4. Британский флаг – 8 лучей (рис. 1.5).



Рис. 1.5. Британский флаг

У Снежинки вырастает один или два боковых луча. Эти ветви уже существенно меньше остальных, но это обычно не мешает – все-таки редко бывает, что последними рассматриваем самые сложные аспекты проблемы.

3.5. Нейрон – неопределенное число лучей (рис. 1.6).

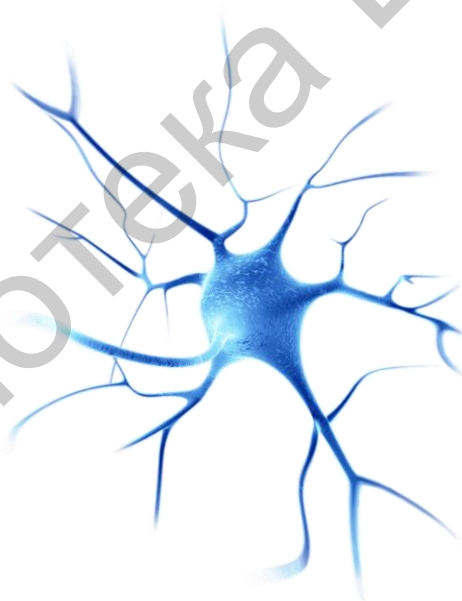


Рис. 1.6. Нейрон

Если все предыдущие формы связаны эволюционной цепочкой, то эта стоит особняком. Это – настоящий дикорастущий куст, никакого формирования кроны. Здесь получают наиболее красивые карты.

В построении mindmap очень хорошо проявляется известный психологический закон 7 ± 2 , описывающий число понятий, которыми может одновременно оперировать мозг человека. Таким образом, границы карты совпадают с естественными ограничениями листа, больше ветвей просто трудно разместить.

3.6. Сколько уровней у mindmap?

Обычно mindmap имеет не более четырех уровней, но больше одного (т. е. ветки раздваиваются). Можно подсчитать, сколько будет веточек (Британский флаг, четыре уровня, каждая ветка раздваивается): $8 \cdot 2^3 = 64$, а всего веток, включая корень, будет $8 \cdot (2^4 - 1) + 1 = 121$. Хотя на практике редко получается такой большой mindmap. Обычно несколько веток имеет по паре уровней, а только отдельные разрастаются.

4. *Метры + килограммы = ?, или однородность.* Очень важным свойством (и одновременно ограничением) является то, что все ветви в принципе однородны, нет разных обозначений для качественно различных сущностей. То есть на одной ветви могут висеть килограммы, а на другой – метры. Это сразу накладывает ограничение на характер информации – пытаться описать бизнес-процесс или структуру базы данных с помощью mindmap бесполезно, лучше использовать специализированные графические нотации – IDEF0, DFD, UML. На mindmap хорошо ложится первозданный беспорядок при первичном анализе или что-то, состоящее из однородных элементов, например план текста или выступления. Обозначаем темы, основные мысли и примеры – и пишем (или выступаем). Обозримость карты позволяет легко переходить от одной части к другой, не теряя из вида целого.

Примеры mindmap

На рис. 1.7 – 1.11 приведены примеры интеллектуальных карт. Как можно увидеть, спектр их применения практически безграничен.



Рис. 1.7. Самопрезентация при помощи mindmap

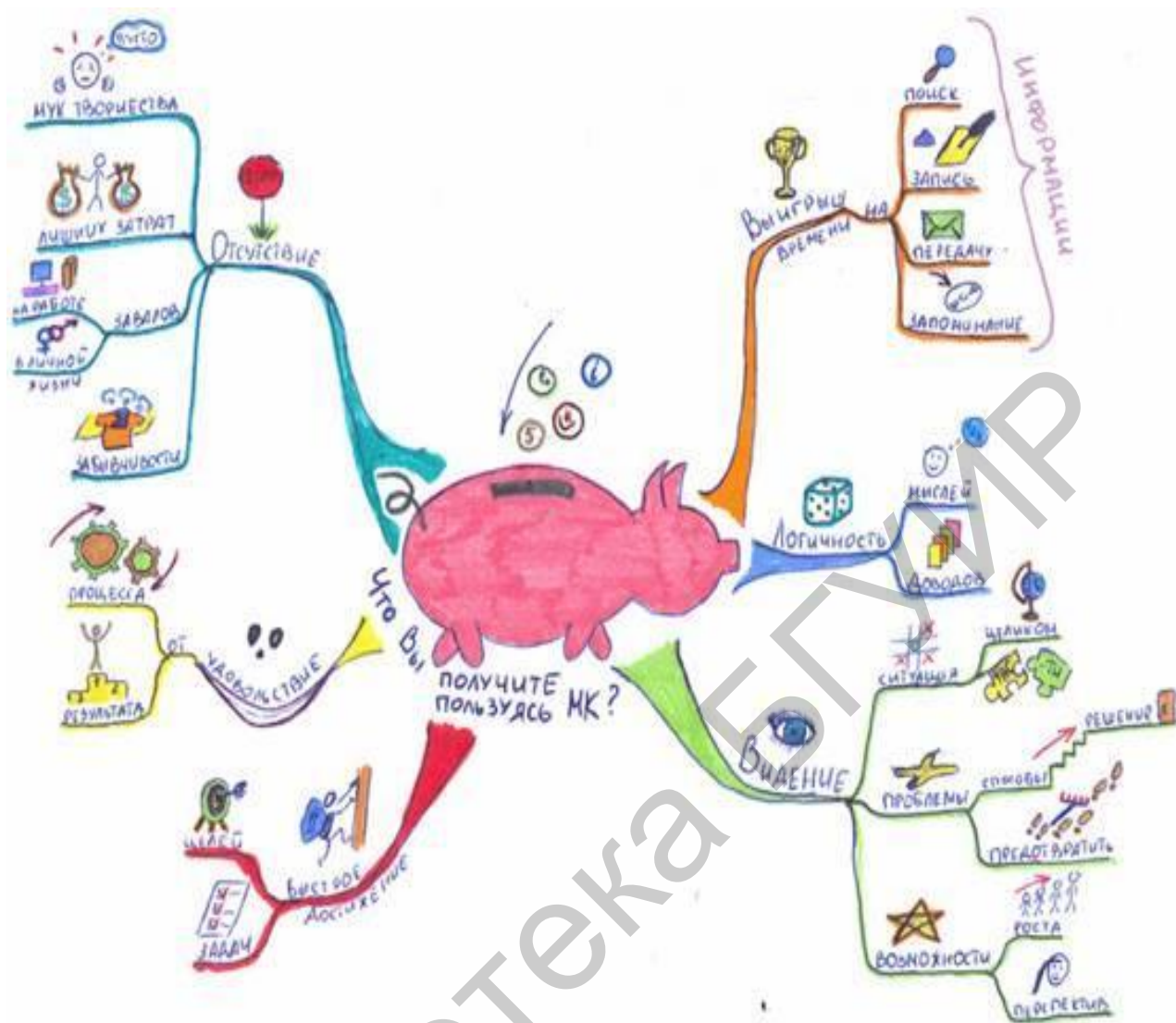


Рис. 1.8. Результат использования интеллект-карт



Рис. 1.9. Интеллект-карта «Mindmap in the classroom»

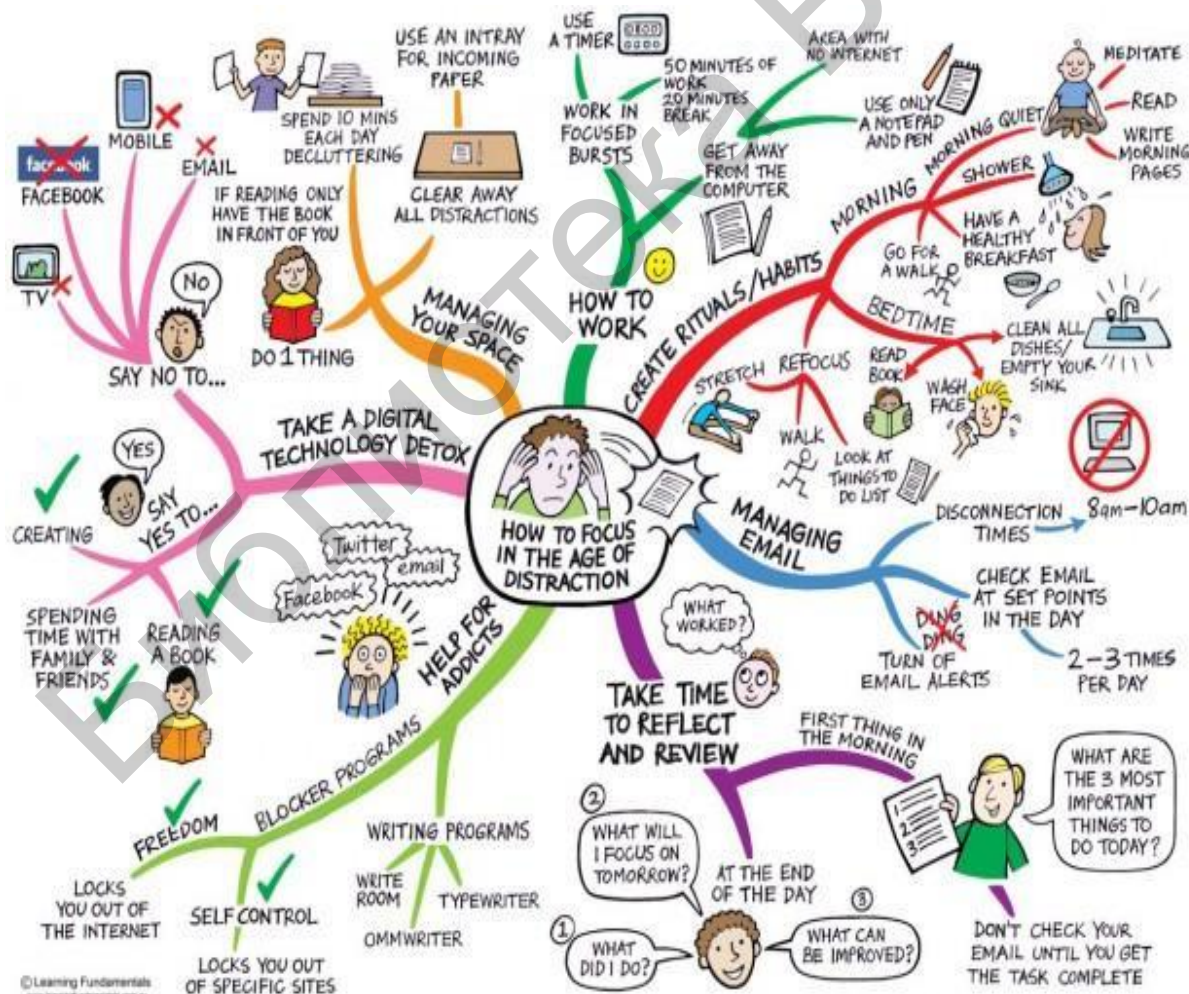


Рис. 1.10. Интеллект-карта «How to focus in the age of distraction»

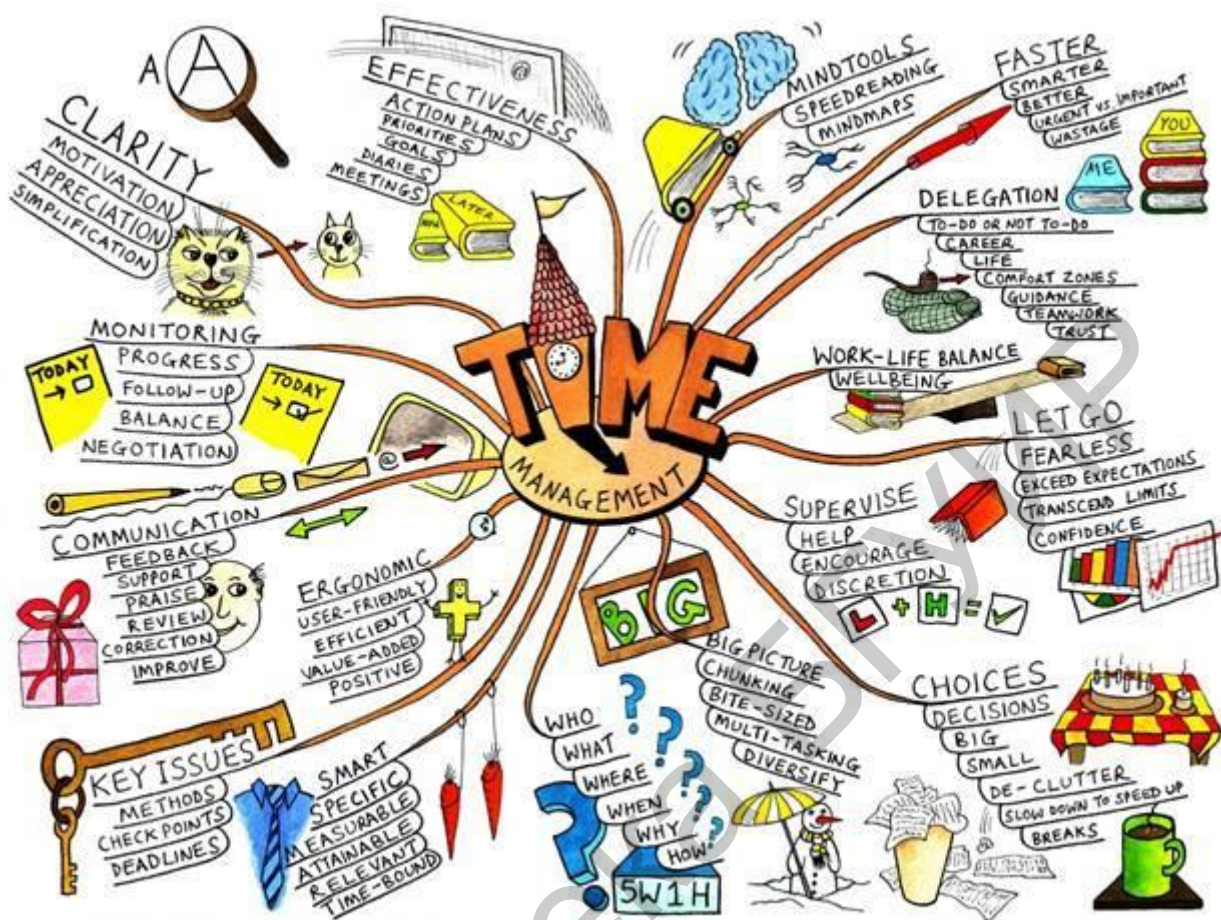


Рис. 1.11. Интеллект-карта «Time management»

Обзор программы XMind

XMind 6 – это программа для визуализации своих идей, а точнее, для составления различных диаграмм. При помощи XMind можно создать fishbone-диаграммы (диаграммы Исикавы), древовидные или логические диаграммы, а также различные таблицы. Все это поможет вам составить наглядный план и четко передать идеи своего проекта другим. В программе уже собраны шаблоны различных диаграмм, которые можно легко и просто отредактировать, и заменить, вставив свой текст и картинки. С XMind также можно создавать качественные презентации и наглядные схемы. Интерфейс программы детально настраивается, достаточно потратить несколько минут, чтобы все «подогнать» под себя. Есть в наличии русский интерфейс, но если ваша система не русскоязычная, то предстоит вручную отредактировать специальный файл программы. Готовую работу можно сохранить в удобном формате, а также экспортировать в Microsoft Word.

Особенности программы:

1. В свои диаграммы вы сможете добавлять маркеры, ярлыки, текстовые пометки, ссылки, аудио-файлы и изображения.

2. Подборка различного клипарта (символы, флаги, смайлики, номера и др.).

3. Экспорт в форматы Html, PNG, GIF, JPEG, BMP. Быстрое создание диаграмм из шаблонов.

4. Создание диаграмм, презентаций и схем для наглядной демонстрации своих идей.

5. Если программа не на русском языке, то в файле «XMind.ini», который находится в папке программы, необходимо дописать в начале текста

–nl

ru

ЗАДАНИЕ

Составить mindmap на заданную тему (варианты тем приведены в табл. 1.1).

Таблица 1.1

Темы mindmap

Номер варианта	Тема mindmap
1	Свое резюме
2	Виденье своей карьеры
3	Здоровый образ жизни
4	Мобильный телефон
5	Чтение
6	Правильное питание
7	Спорт
8	Культура поведения
9	Вода
10	Компьютер
11	Технологический прогресс
12	Транспорт
13	Оружие
14	Место жительства
15	Строительство

ДОМАШНЕЕ ЗАДАНИЕ

Составить mindmap по лекции на выбор студентов.

Часть 2

ЗНАКОМСТВО С МЕТОДИКОЙ BRAINSTORM И МОРФОЛОГИЧЕСКОГО АНАЛИЗА

1. BRAINSORM

Мозговой штурм, или brainstorm, – оперативный метод решения проблемы на основе стимулирования творческой активности, при котором участникам обсуждения предлагают высказывать как можно большее количество вариантов решения, в том числе самых невероятных. Затем из общего числа высказанных идей отбирают наиболее удачные, которые могут быть использованы на практике. Является методом *экспертной оценки*. В развитом виде предполагает синхронизацию действий участников в соответствии с распознаваемой ими схемой (образом) оцениваемого процесса (process pattern recognition).

Метод мозгового штурма (brainstorm) был изобретен в 30-е гг. XX в. Алексом Осборном. В настоящее время существует более 50 модификаций brainstorm.

Суть метода заключается в разделении этапов выдвижения и анализа идей при решении проблем.

Цель мозгового штурма – выдвижение как можно большего числа разнообразных идей.

Во время сеанса brainstorm происходит так называемая цепная реакция идей, приводящая к интеллектуальному взрыву. Новые идеи возникают подобно электрической искре при контакте с мыслями и идеями других людей.

Зона оптимальности использования brainstorm:

1. Задачи, когда мы еще не обладаем большими знаниями об объекте исследования и нужно выявить направления решения проблемы. На этом этапе мы пока не можем применить логику. Допустим, проблема возникла недавно.

2. Задачи на проектирование новых товаров, идей рекламы.

3. «Обратные» задачи – как ухудшить товар, процесс, услугу. Решаются эти задачи для того, чтобы после обратного brainstorm найти «лишние звенья» и исключить из бизнес-процессов ненужные и наносящие ущерб процедуры.

Рекомендации по проведению brainstorm

В brainstorm участвуют ведущий и две группы людей: генераторы идей и критики-аналитики. Всего – не более 4–8 человек разных специальностей.

Требования к участнику – генератору идей

Задача генератора идей – непрерывно выдвигать идеи по поставленной проблеме или задаче.

Требования к генераторам: выдвижение по широкому спектру тем большого количества идей, основанных на новых принципах; перенос идей из различных областей; использование ярких неожиданных аналогий. Возможно использование ранее выдвинутых идей другим генератором. В основе brainstorm лежит идея непрерывного поиска новых, лучших идей – оптимизм. Недопустимо включать в группу генераторов скептиков и критиков.

Отбор генераторов идей и формирование группы:

1. По психологическим признакам творческой личности, личные наблюдения, тесты. Типичный генератор активно выдвигает идеи по любой предложенной теме, задаче в присутствии третьих лиц и при наличии критики («фонтан идей»). Теневой генератор – активно выдвигает идеи индивидуально. Инертный генератор – положительная установка на творчество, но нет опыта, низкий уровень притязаний.

2. В ходе учебного brainstorm.

3. Группе людей рассказывается о задачах генераторов идей и аналитиков. Затем каждый пишет, в какой группе он хотел бы видеть себя и других присутствующих.

4. Разнообразие опыта (желательно наличие людей с разным образованием – экономистов, снабженцев, конструкторов и т. д., а также людей из совсем других отраслей – повар, проводник, парикмахер и т. д.)

5. Желательно иметь в группе генераторов женщин и мужчин.

6. Недопустимо наличие в группе генераторов идей отношений служебной подчиненности.

Требования к участнику: критик-аналитик

Задача критика-аналитика: выявление рационального зерна в каждой предложенной к анализу идее на основе глубокого знания теории и практики, понимания специфики проблемы (задачи), способности к обобщению. Ему должна быть присуща вера в то, что лучшая идея – это та, которая анализируется в данный момент, оптимизм. Хорошие аналитики могут выявить новые принципы решения задачи после классификации принципов, выдвинутых генераторами. Очень часто самое ценное в brainstorm – новое направление поиска, а не конкретное решение.

Требования к участнику: ведущий

Ведущий обеспечивает качество регулированием хода brainstorm.

1. Отвечает за процедуру и регламент работы (соблюдение правил). Прерывает неконструктивное поведение участников: «перетягивание каната», «павлиний хвост» и др.

2. Обеспечивает психологическую поддержку участников, атмосферу активности и доброжелательности (должен уметь мгновенно положительно оценить любую идею).

3. Управляет процессом поиска идей (необходимо постоянно уточнять формулировки задачи, расширять поле поиска, выделять новые направления и аспекты решений, задавать новые области поиска решений для участников).

Правила для участников brainstorm (мозгового штурма):

1. Стремиться высказывать максимальное число идей. Отдавайте предпочтение количеству, а не качеству идей. Высказывайте свои идеи короткими предложениями – не более 15 с.

2. Абсолютно запрещена критика предложенных идей, неодобрительные замечания, иронические реплики, «ядовитые» шутки (например: А что скажет директор? Для практики это не годится! Это же чепуха! Так еще никогда не делали!...)

3. Отдавайте предпочтение не логическому мышлению, а озарениям и безграничной фантазии в самых разных направлениях. Приветствуются любые идеи.

4. Очень способствуют продуктивному мышлению шутки, каламбуры, юмор и смех. Поддерживайте и создавайте такую обстановку.

5. Стремиться развивать, комбинировать и улучшать высказанные ранее идеи, получать от них новые ассоциативные идеи.

6. Обеспечивайте между участниками brainstorm свободные, дружеские и демократические отношения. Не допускается после сеанса зло шутить над идеями других участников.

Организация проведения мозгового штурма (brainstorm)

Приглашать на участие в brainstorm надо заранее. Нежелательно сообщать формулировку задачи.

Помещение: закрытое на ключ, без посторонних шумов и без телефонов. Обстановка должны быть теплой и уютной. Возможна негромкая фоновая музыка. В помещении не должно быть никаких посторонних людей.

Стол: круглый. Можно чай или кофе, напитки, но без съестного.

Средства повышения эффективности brainstorm:

1. Показ перед мозговым штурмом (brainstorm) короткометражного фильма, заставляющего забыть заботы дня, или фильма по теме задачи.

2. Показ натурального образца, макета или эскиза объекта, который надо улучшить.

3. Показ на экране случайных предметов слов (существительных, глаголов).

Оформление результатов brainstorm

Запись всех идей стенографистом или каждый участник записывает свои идеи. Диктофон, магнитофон.

Этапы мозгового штурма (brainstorm)

I. Этап генерации идей

Формулировка задачи. Представить проблему надо в таком виде, чтобы она не вызывала психологических барьеров (без специальных терминов), снять уровень ответственности за свои предложения. При представлении проблемы должны быть четко сформулированы два момента:

1. Что в итоге желательно получить или иметь.
2. Что мешает получению желаемого.

Виды формулировки проблемы:

1. В виде формулировки проблемы-аналога.
2. Формулирование проблемы в обобщенном виде.
3. Проблема в том виде, в котором она дана (самый слабый вид формулировки).

Проведение мозгового штурма (brainstorm). Обычно фиксируются следующие фазы мозгового штурма (brainstorm):

- включение;
- наполнение;
- прорыв (новое оригинальное направление);
- индукция (идеи после окончания штурма) (иногда они «выскакивают»

в течение нескольких дней).

Перерыв 5 мин.

Составление отредактированного списка идей – предварительный анализ и индукция: наиболее приемлемые и легко реализуемые идеи для решаемой задачи; наиболее эффективные идеи; перспективные и т. д.

II. Этап анализа идей

Процедуры этапа анализа для каждой идеи:

1. Уяснение идеи в виде, сформулированном на этапе генерации.
2. Обобщение идеи. Выявление принципа. (Ведение классификации принципов, дополнение классификации).
3. Оценка перспективы принципа (обязателен специалист).
4. Наполнение принципа конкретным содержанием (желателен специалист).
5. Развитие полученной идеи (не для всех идей).

Желательно вывести идеи на одинаковый уровень общности, например:

1. Формулировка целей и целей альтернатив.
2. Обобщенные (функциональные) принципы достижения цели.
3. Структурные принципы реализации функции.
4. Физические принципы реализации структуры.
5. Технические устройства, осуществляющие каждый принцип.

Достоинства brainstorm:

1. Легкость освоения и простота в обращении.
2. Незначительные затраты времени на проведение.

3. Универсальность метода.
4. Наиболее эффективен при решении организационных проблем, а также технических задач невысокого уровня сложности.

Недостатки brainstorm:

1. Решение относительно простых задач.
2. Отсутствие критериев, дающих приоритетные направления выдвижения идей.
3. Нет гарантии нахождения сильных идей.

ЗАДАНИЕ BRAINSTORM

Разделиться на группы по 4–6 человек и получить задание согласно варианту (табл. 1.2).

Таблица 1.2

Темы заданий brainstorm

Номер варианта	Задание
1	Вымышленный мир
2	Вымышленное существо
3	Где провести выходные?
4	Придумать минимум 20 способов использования канцелярской скрепки
5	Критерии выбора площадки для офиса
6	Варианты идей для решения проблемы «Как стащить тяжелую лодку в воду?»
7	Как заработать 1000 евро?
8	Почему дела не «идут»?
9	Какой бизнес открыть?

2. МОРФОЛОГИЧЕСКИЙ АНАЛИЗ

Морфологический анализ – это метод систематизации перебора вариантов всех теоретически возможных решений, основанный на анализе структуры объекта.

Как таковой *морфологический анализ* придумал американский астроном швейцарского происхождения Фриц Цвикки. Именно с помощью этого метода Цвикки в 30-х гг. XX в. предсказал существование нейтронных звезд, а также предположил наличие во Вселенной так называемых «адских звезд», описание которых чудесным образом оказалось крайне схожим с открытыми сорок лет спустя черными дырами. Таким образом, американец, по сути, открыл с помощью морфологического анализа черные дыры. Ф. Цвикки выпустил книгу «*Морфологическая астрономия*», где описал возможности прогнозирования астрономических открытий с помощью разработанного им морфологического метода. Однако в дальнейшем широкого применения в астрономии морфологический анализ так и не нашел, зато замечательно пригодился разного рода изобретателям и фантастам.

Суть морфологического анализа

Морфологический анализ заключается в систематическом обзоре всех мыслимых вариантов данного явления или объекта. Не больше, но и не меньше. Как это работает на практике? Мы выбираем какой-либо объект и составляем список его основных характеристик. Далее для каждой из характеристик мы перечисляем все возможные варианты. Таким образом, комбинируя эти самые варианты, мы получаем в итоге огромное множество новых, еще нигде не фигурировавших объектов.

Итак, выберем для примера компьютер. На первом этапе мы перечисляем его характеристики: тип блока питания, производительность, способ управления, энергопотребление, габаритные размеры и т. д. Чем больше выделено характеристик на этом этапе, тем большее количество вариантов может быть получено на выходе.

Далее для каждой из характеристик составляем список всех мыслимых и немыслимых вариантов, необязательно относящихся к данному объекту. Например:

1. *Блок питания*: электрический, паровой, механический, гидравлический, внутреннего сгорания, ядерный и т. д.

2. *Производительность* (в сравнении с лучшей моделью на данный момент): низкая, средняя, высокая, сверхвысокая и т. д.

3. *Способ управления*: ручной, голосовой, жестовый, мысленными командами, интуитивный, аппарат сам предугадывает желания.

4. *Энергопотребление* (в сравнении с моделями данного времени): сверхнизкое, низкое, среднее, высокое, сверхвысокое.

5. *Размеры* (в сравнении с современными моделями): сверхмалые, малые, средние, большие, сверхбольшие.

Таким образом, комбинируя полученные варианты, можно представить себе, например, управляемый мыслью сверхмалый компьютер с чрезвычайно низким энергопотреблением (как, например, от биотоков самого человека). Или как вам, например, компьютер, размером с планету с фантастической производительностью, использующий для работы энергию внутренних процессов занятого им небесного тела? В общем, вариантов может быть очень много, вам остается лишь выбрать наиболее подходящий. Здесь основное значение имеет первоначальное количество характеристик в первом шаге и оригинальность вариантов их исполнения во втором. Если составленная таким образом *морфологическая таблица* получилась обширной и полной, сформировать из нее оригинальное фантастическое допущение не составит никакого труда.

Алгоритм последовательности действий при морфологическом анализе:

1. Точно сформулировать проблему.
2. Определить важнейшие элементы.
3. Определить варианты исполнения элементов.
4. Занести их в таблицу.
5. Оценить все имеющиеся в таблице варианты.
6. Выбрать оптимальный вариант.

Пример схематичного морфологического анализа можно посмотреть на рис. 1.12.

Морфологический анализ

Метод поиска новых идей, основанный на анализе структуры объекта

Правила:

Формируется проблема.
Указываются основные параметры объекта.
Указываются варианты исполнения параметров.
Рассматриваются все возможные сочетания.

Задача:

Найти новый вид тары для соков



Рис. 1.12. Морфологический анализ

ЗАДАНИЕ ПО МОРФОЛОГИЧЕСКОМУ АНАЛИЗУ

Персонально (с учетом варианта) выполнить задание по составлению таблицы и выбору лучших вариантов (табл. 1.3).

Таблица 1.3

Темы заданий на морфологический анализ

Номер варианта	Тема задания
1	Апгрейды автоматического стрелкового оружия в игре
2	Свойства холодного оружия в игре
3	Реализация возможности отдыха игрового персонажа (условия, необходимые предметы, приносимая польза)
4	Игровой транспорт в космическом шутере
5	Приспособления для быстрого перемещения по миру в фэнтезийной RPG
6	Использование рисования в обучающих играх
7	Боевые классы для кооперативной игры (тип урона, оружие, особенности, слабые стороны, преимущество, взаимодействие с другими классами)
8	Система игрового крафта
9	Материалы для создания одежды и обмундирования в игре (материал, описание, свойства)
10	Реализация приготовления блюд в игре
11	Механизм охоты в играх с открытым миром (условия, поведение тех, на кого охотится игрок, полезность охоты, возможные трофеи и условия их получения, использование трофеев)
12	Альтернативный вид рогатого шлема в игре Skyrim

Лабораторная работа №2

СОСТАВЛЕНИЕ АНАЛИТИЧЕСКОГО ОБЗОРА КОМПЬЮТЕРНОЙ ИГРЫ

Жанр

Можно с самого начала представлять в мельчайших деталях свою готовую игру, а можно по ходу разработки додумывать и сюжет, и стиль, и особенности игры. В этом деле не обязательна излишняя точность, но, как минимум, нужно задать направление развития нашего игрового проекта. Жанр игры необходимо выбрать в самом начале в обязательном порядке. Жанр и будет основным направлением развития игры.

Классификация игр по жанрам

Основной критерий при определении жанра компьютерной игры – действие, чаще всего совершаемое в играх этого жанра.

Все игры разделены на три большие группы (рис. 2.1):

1. Игры информации (ролевые игры): главное – получение информации во всех ее проявлениях. В таких играх иногда присутствует и планирование, и динамика, но информация в них намного важнее. В таких играх можно жить, вживаться в роль героя, в качестве главных достоинств выставляют атмосферу, сюжет, игровой мир.

2. Игры действий: главное – движения, которые необходимо совершать, управляя каким-либо телом (человеческим или гуманоидным) либо техническим средством. Это наиболее динамичные игры. Именно игры действий принято хвалить за то, что они развивают скорость реакции.

3. Игры контроля: главное – планирование событий и управление для достижения преимущества в дальнейшем. Сюда попадают все виды стратегий, различные экономические игры, варгеймы, тактики.

Выделены 15 основных элементов геймплея, из которых состоит любая игра (табл. 2.1). Жанр зависит от их совокупности.

Таблица 2.1

Элементы геймплея		
Игры информации	Игры действий	Игры контроля
Обучение Загадки Общение Роль Изучение	Собирание Уклонение Уничтожение Соревнование Вожделение	Забота Создание Контроль Тактика План

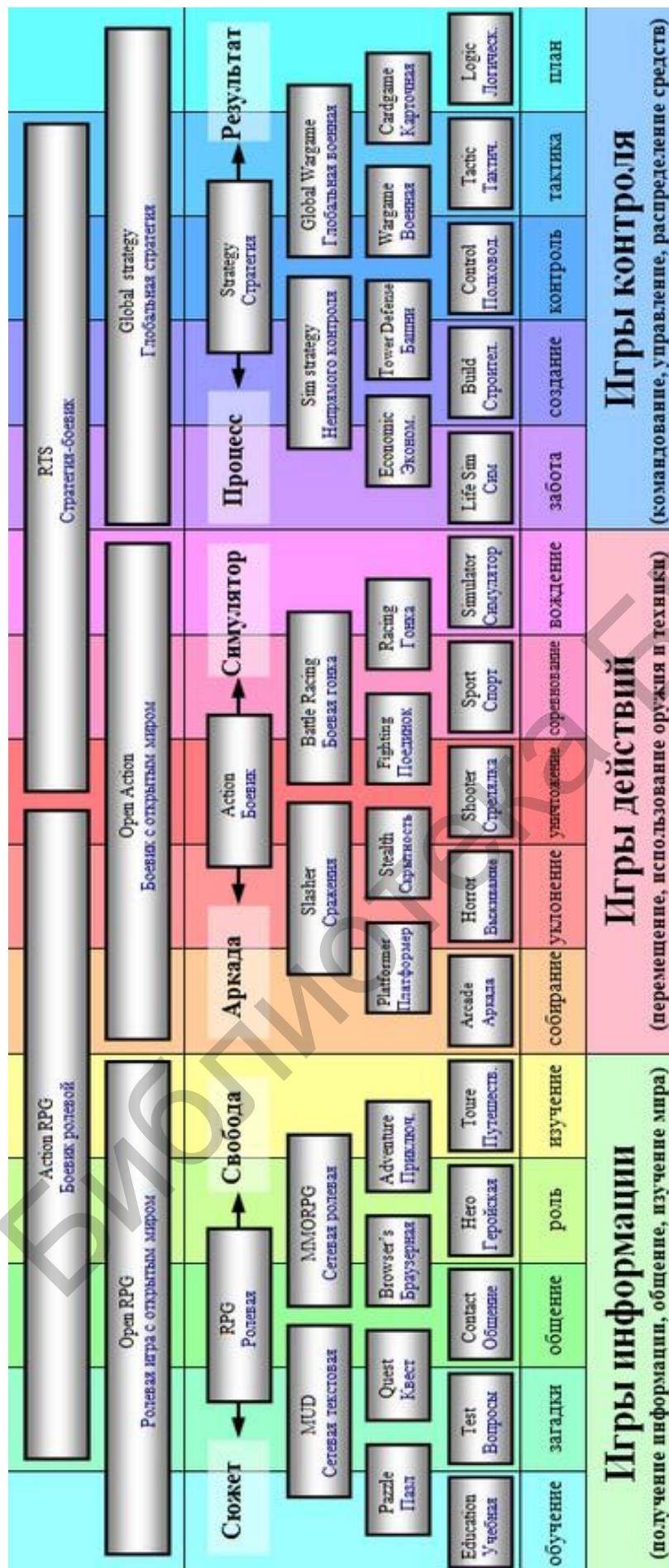


Рис. 2.1.1. Классификация игр по жанрам

Все игры внутри группы очень похожи между собой, но одновременно с этим имеются игры с различными особенностями. Эталоны групп игр находятся на золотой середине, а жанры со все большими особенностями – все дальше от центра.

Жанры и эталоны групп:

1. Игры информации (ролевые игры): на одной границе группы – прямой сюжет (где имеется единственный правильный путь и невозможно отойти хотя бы на шаг от этого пути), на другой границе противоположность – абсолютная свобода (огромный открытый мир, свобода действий и полное отсутствие сюжета). Эталон – RPG (RolePlaying Game) – «ролевая игра».

2. Игры действий: на одной границе группы – аркадность (простота в освоении, нереальность происходящего), на другой границе – противоположность – симуляторность (сложность в освоении, реалистичность). Эталон – Action (игра-боевик).

3. Игры контроля: на одной границе группы – игра во имя результата и достижения цели, на другой границе противоположность – игра во имя самого процесса игры. Все жанры группы лежат между этими двумя крайностями. Эталон – Strategy (обычная локальная стратегия).

На рис. 2.2 проиллюстрированы три группы игровых жанров и переходы между ними.



Рис. 2.2. Переход между игровыми жанрами

Чтобы выбрать (определить) жанр игры, смотрим на рис. 2.2, как на мишень, и выбираем необходимые игровые геймплейные элементы. Некоторые элементы подарят нам высокую популярность среди игроков (уничтожение, соревнование, герой, забота), некоторые – преданных, но придиричивых фанатов (тактика, контроль, уклонение), а некоторые – отсутствие серьезных конкурентов (обучение, логика, путешествие, экономика).

Выбранный жанр можно немного корректировать по ходу работы, но его сущность должна оставаться прежней. Жанр – это своеобразный фундамент всей игры. Если захотите сменить жанр своей игры, то проще будет начать разработку новой игры заново, чем переделывать то, что уже было наработано.

Сеттинг

Игровые жанры определяют основные действия, которые будут совершать игроки в процессе игры. Сеттинг же определяет место совершения этих действий (тип вымышленного игрового мира) и время совершения этих действий (историческую эпоху).

Сеттинг – это принадлежность игры к какой-то сюжетной теме или к определенному виртуальному миру, декорации, в которых происходят игровые события. В среде компьютерных игр сформировалось несколько наиболее популярных сеттингов.

Вымышленные миры:

1. **Реальный мир:** игровой мир практически не отличается от нашего.
2. **Параллельные миры:** реальный мир с переходами в искаженные миры.
3. **Альтернативная история:** другой исход исторических событий и, соответственно, другое развитие мира.
4. **Фэнтези:** сказочный мир с наличием магии.
5. **Геройская мифология:** сверхсущества, супергерои, мутанты, гибриды, полубоги.
6. **Христианская мифология:** ангелы, демоны, наличие рая и ада.
7. **Современная мифология:** нашествие зомби/инопланетян, пост-апокалипсис.

Исторические эпохи:

1. **Зарождение жизни:** простейшие организмы, игры на клеточном уровне.
2. **Доисторические времена:** эра динозавров, пещерные люди.
3. **Средневековье:** рыцари, крестовые походы, инквизиция, междоусобные войны.
4. **Эпоха колонизации:** морские путешествия, исследование новых земель.
5. **Эпоха индустриализации:** XVIII–XIX вв., стимпанк.
6. **Прошедшие войны:** Первая и Вторая мировые войны, война во Вьетнаме, война в Афганистане, локальные стычки, события в Ираке и т. д. (сеттинг заключен между началом и концом войны, он не выходит за ее пределы).
7. **Наше время.**
8. **Информационная эпоха:** киберпанк, антиутопии будущего.
9. **Освоение космоса:** Sci-Fi, космические путешествия.
10. **Эволюция:** наличие нескольких эпох в одной игре.

Создание игры в популярном сеттинге обеспечивает ее собственную популярность, игроки чувствуют себя уютно и комфортно в уже знакомом мире. Некоторые игры создаются в своих уникальных сеттингах или в необычных сочетаниях стандартных тем. Такие игры менее популярны, но тем не менее они имеют свою аудиторию особых игроков, которые терпеть не могут шаблонность и однообразие.

Целевая аудитория

Многие разработчики затрудняются ответить на вопрос: *для кого вы это создаете?* Как показывают опросы и опыт, большинство ответов на подобные вопросы звучит слишком расплывчато или вообще является догадками.

В первую очередь вы создаете игру для конкретной целевой аудитории. Если на ранней стадии разработки вы неверно определили вашу ЦА, то весь процесс пойдет в неверном направлении. Когда игры не оправдывают ожиданий игроков, можно смело обвинять геймдизайнеров, которые с самого начала ошиблись в выборе ЦА или вообще проигнорировали этот процесс.

Что нужно учитывать при выборе ЦА:

1. Демографические данные возможных игроков (место жительства, пол, возраст, образование, уровень дохода, потребности).
2. Образ жизни возможных игроков (принятие решений, способность транжирить деньги).
3. Интересы предполагаемых игроков.
4. Личные цели предполагаемых игроков (чего хотят добиться в жизни).
5. Эмоциональность предполагаемых игроков (сила испытываемых эмоций, реакция на негативные/позитивные события).
6. Поведение предполагаемых игроков (как и когда игроки готовы тратить деньги, как это относится к вашему будущему проекту, как игроки относятся к скидкам, как взаимодействуют с другими игроками и взаимодействуют ли вообще).
7. Что предполагаемые игроки хотят получить от вашего проекта (нужно ли им игровое сообщество, нужны ли им дополнительные сервисы и т. д.).

Чаще всего игроков разделяют на казуальных, мидкорных и хардкорных (табл. 2.2).

Таблица 2.2

Распределение игроков по категориям

Параметр	Казуальный		Мидкорный		Хардкорный	
	М	Ж	М	Ж	М	Ж
Соотношение	40%	60%	55%	45%	72%	28%
Возраст	≈18	≈35	≈25	≈28	≈20	≈25
Устройства	смартфон / приставки		смартфон / приставки		смартфон / приставки / PC	
Ежедневно играют	20–60 минут		30–120 минут		120–300 минут	
Цель игры	Убить время		Удовольствие и убить время		Удовольствие и прокачка навыков	
Жанры	- Азартные - Настольные	- Приключения - Головоломки	- РПГ - Экшн	- Стратегии - Фермы	- MMO - Шутеры - Гонки	- РПГ - Экшн - MMO

Также разделение пользователей происходит по платформам, которые выбираются тем или иным игроком. Примерная статистика приведена на рис. 2.3.

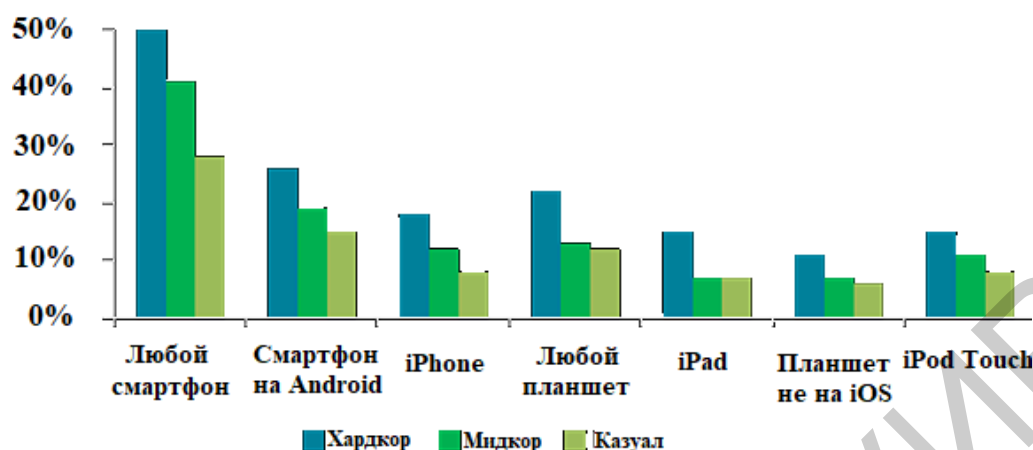


Рис. 2.3. Распределение игроков по устройствам

С появлением внутриигровых покупок игроков стали разделять и по уровню финансовых возможностей (рис. 2.4).

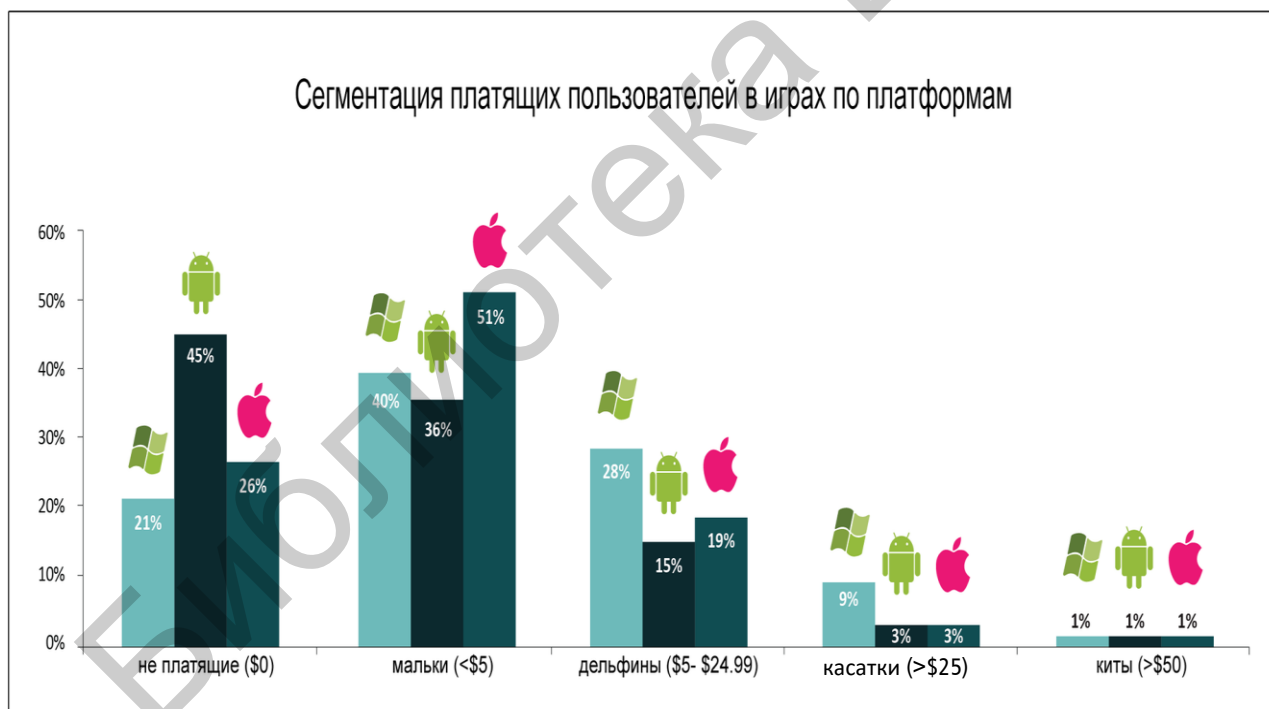


Рис. 2.4. Сегментация пользователей по платформам

Давайте попробуем «оживить» пользователей и разделить их на группы по количеству внутриигровых покупок (рис. 2.5, 2.6).

Фактическая оплата	 Мужчины	 Женщины	 Средний возраст	 Недельная вовлеченность мобильными играми	 Недельная вовлеченность всеми играми
не платят	35%	65%	36.6	4.1 часа	7.2 часа
платят мало	53%	47%	32.5	4.9 часа	9.9 часа
платят много	66%	34%	30.2	11.8 часа	26.5 часа
в среднем	44%	56%	34.6	—	—

Рис. 2.5. Обзор возможностей оплаты разными пластами игроков

ИГРОВОЕ ВРЕМЯ НА ВСЕХ ТИПАХ УСТРОЙСТВ

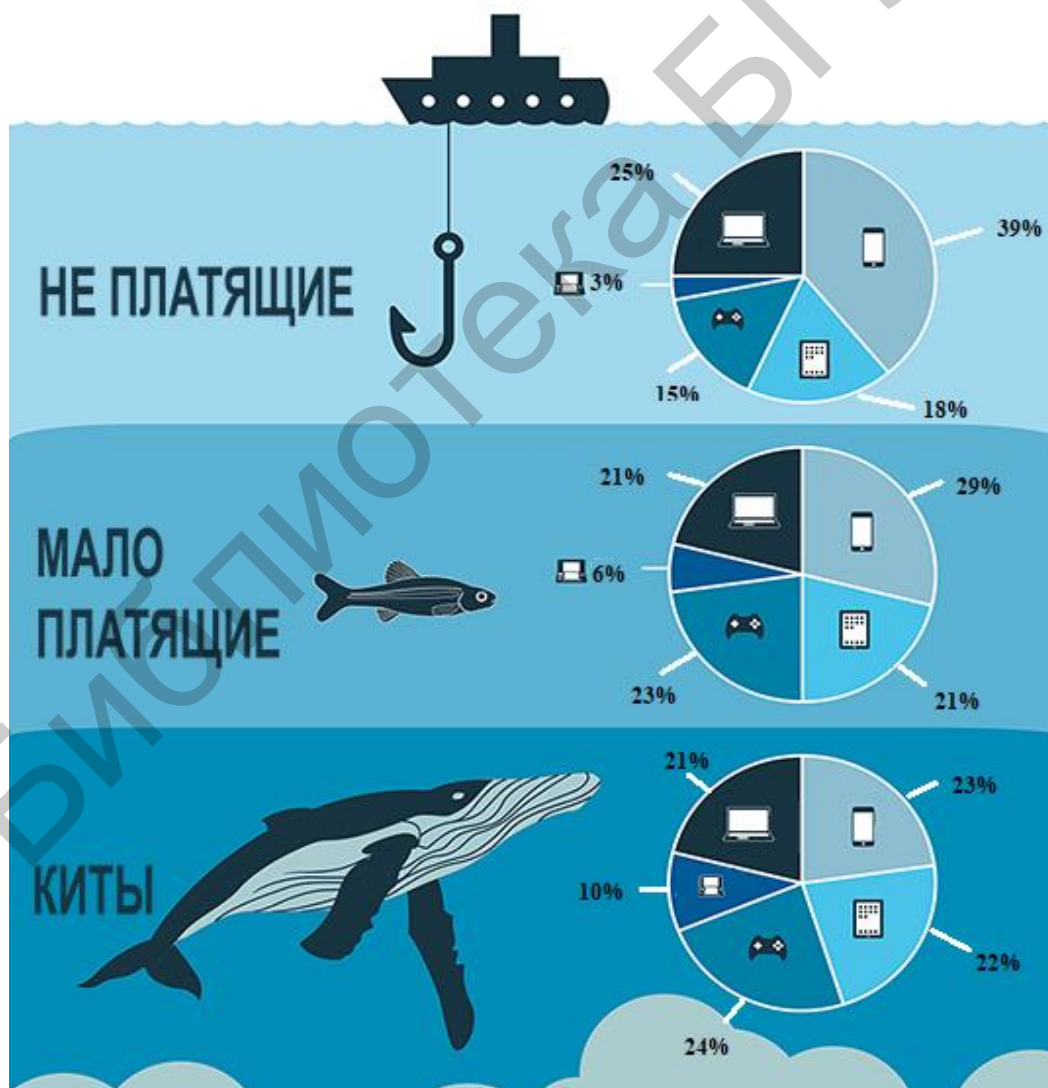


Рис. 2.6. Соотношение игрового времени на всех платформах

В настоящее время самая известная проблема – это уверенность разработчиков, что их игра подойдет для всех. Можно встретить много игр, которые завоевали популярность среди игровых критиков, но так и не смогли снискать популярность среди игроков.

Как принимать решения, влияющие непосредственно на опыт игрока, без понимания, для кого ты собственно их принимаешь? Для решения этой задачи нужно обратиться к потребностям пользователей. Если заранее продумать, чего игроки хотят от вашего будущего проекта, правильные решения не заставят себя ждать. На ЦА также влияют выбор жанра и сеттинга. Механизм выбора/определения ЦА только один: учесть потребности разных игроков и понять, кому из них будет интересен ваш проект.

Список элементов игры – описание своеобразных «кирпичей», из которых выстраивается проект. Это основные игровые механики, сущности и их взаимодействия.

Список циклов в игре

Игровой цикл – принцип, согласно которому геймдизайнеры задают главный элемент игровой механики, который определяет фундаментальный опыт игрока.

Один игровой цикл представляет собой действие игрока, результат этого действия в игровом мире, реакцию игрока на результат и запрос игры на повторение нового действия.

Игрок постоянно совершает действия основного цикла игры. Например, в ролевой игре он убивает монстров, за это его персонаж получает как опыт, так и выпадающие из монстров предметы. Это позволяет персонажу стать более сильным, что дает возможность побеждать еще более сильных монстров и т. д.

В игре может быть несколько циклов, связанных между собой и реализующих крупные составные механики, это и описывается в списке циклов игры.

Метагейм (от греч. meta – «за пределами», «после» и англ. game – «игра»), или метагейминг, метаигра, – использование в игре того, что относится к игрокам, а не к их персонажам.

Метаигровая информация – информация, полученная неигровыми способами.

Метагеймовые рассуждения – мысли игрока, а не его персонажа.

Метагеймовый комментарий – комментарий, сделанный «за кадром» игроком или мастером для других участников игры.

Метагеймовое решение – принятое игроком исходя из метагеймовых соображений, а не из логики персонажа, нередко в ущерб отыгрышу.

В форумных ролевых играх слова «метагейм» и «оффтопик» часто употребляются как синонимы.

Примером метаигры могут служить коллекционные карточные игры: многие игроки тратят огромное количество времени, сил и денег на поиск нужных карт и на составление игровых колод.

Метагейм может принимать следующие формы:

1. Использование знания игромеханики (нередко в ущерб отыгрышу) – например, использование боевой тактики, направленной на слабости монстра, известные игроку по прошлым играм или почерпнутые напрямую из «монстрятника» (в «AdvancedBestiary» GRP рекомендуют менять описание монстра с сохранением его оцифровки, фактически обманывая игроков).

2. Использование сеттинговых сведений, неизвестных персонажу из-за его образования, воспитания, принадлежности к той или иной группе населения, но известных игроку опять-таки по прошлым играм или почерпнутых из чтения сеттинговых дополнений. Такие сведения, строго говоря, метаигровыми не являются, в отличие от способа их получения.

3. Знание модуля: некоторые модули содержат сюжетные повороты, ловушки или загадки, требующие нетривиального подхода к их распутыванию, но легко проходимые теми, кто про них знает. Многие модули содержат предупредительные надписи о том, что некоторые разделы читать категорически не следует тем, кто собирается по этому модулю играть, – и дело тут не в том, чтобы не дать игрокам «победить» мастера, а в том, что получаемое от прохождения модуля удовольствие идет от самого процесса преодоления препятствий, а не от нажатия нужных кнопок.

4. «Подглядывание» как реакция персонажа на события, свидетелем которых он(а) не был(а). Из-за сложности «развидеть» увиденное и забыть услышанное мастера в настольных играх удаляют игроков в некоторых сценах «на кухню» или закрывают чужие ветки на форумной игре. Если объемы конфиденциальной информации невелики, можно обойтись записками или личными сообщениями.

Взгляды на поощрение и подавление метаигрового мышления игрока в разных системах разные, да и то, что метагейм всегда и в любых видах является абсолютным злом – широко распространенное заблуждение. В терминах *Большой модели* выделяется несколько позиций (stance) игроков по восприятию своих персонажей во время игры: актерская – подразумевает погружение и восприятие игрового мира с позиции персонажа; авторская – позволяет игроку наблюдать за своим персонажем несколько отстраненно, «от третьего лица», фокусируясь в первую очередь на действиях и изменениях вокруг персонажа, а не на переживаниях; режиссерская – с восприятием персонажа как части общей сцены, которую игрок контролирует наряду с другими деталями (описанием «декораций», развитием событий, возможно, действиями второстепенных персонажей). Актерский подход не предусматривает никакого метагейма, авторский может быть вырожден до так называемого марионеточного, когда все действия персонажа целесообразны с точки зрения игрока, а режиссерский является метагеймовым по определению и без метагейма невозможен. Практически любая игра, подразумевающая возможность игрока вмешиваться

не только в действия своего персонажа, требует от игрока периодического переключения на режиссерское восприятие игры. Подобный подход является ключевым для «красивой» игры, например, в ситуациях, когда у игрока под контролем находится несколько персонажей и создание сцены зависит от их взаимодействия.

Степень ориентации игры на режиссерский подход можно определять по механизму передачи нарративных прав. В играх, где подобный подход считается ограниченно полезным или вовсе ненужным, у игрока обычно нет возможности получить эти права в отношении действий не своего персонажа либо подобные ситуации строго лимитированы; в играх, где поощряется подобное мышление, такие ограничения обычно более слабы.

Аналитический обзор игры – обзор основных элементов и циклов игры, оценка ее ЦА, стилистики, определение жанра и сеттинга. Обзоры можно использовать для того, чтобы рассмотреть несколько игр одного жанра, выделить интересные механики и использовать их в собственном проекте.

Пример аналитического обзора игры Pitches

Игра жанра спортивный менеджер в реалистичном сеттинге – аналог Football Manager только по теме волейбол, упрощенный для midcore аудитории и в онлайн. Важнейшим отличием игры от игры-конкурента станет то, что core-геймплеем будет выступать не столько стратегическое управление командой, сколько сами волейбольные матчи. Именно решения, принимаемые игроком во время матча, будут иметь решающее значение для результата. Таким образом, в отличие от игр-конкурентов, где основной упор делается на сам менеджмент команды, эта игра будет больше про волейбол, тактику и взаимоотношения спортсменов с тренером, а не просто экономический менеджер в спортивном сеттинге.

Сеттинг: реализм, спортивный менеджер.

Стиль арта: схематичный (рис. 2.7, 2.8).



Рис. 2.7. Арт игры Pitches



Рис. 2.8. Вид игровой сцены игры Pitches

Жанр: sport/strategy.

Описание игры. Игрок управляет волейбольным клубом как главный тренер и владелец. Он подбирает игроков, тренирует их, развивает клубную инфраструктуру, участвует в турнирах, управляет командой во время матчей.

Основная особенность. Основной механикой, вокруг которой будет построен core-геймплей, будет вмешательство игрока в спортивный матч через управление тайм-аутами, которое позволяет оказывать значительное воздействие на эффективность действий команды. В соответствии с правилами волейбола тренер может заказать тайм-аут всего несколько раз за весь матч, при этом волейбольный тайм-аут длится всего 60 с. В игре эти 60 с будут ресурсом, которым может воспользоваться игрок, чтобы внести корректировки в игру, изменить тактику, подбодрить или поругать кого-то из игроков. Каждое действие во время тайм-аута будет иметь свою цену в секундах и игроку будет необходимо тщательно спланировать все, что он хочет сказать и сделать во время тайм-аута. Таким образом, игрок при помощи специального интерфейса определяет, что он будет делать в ближайший тайм-аут и нужный момент, либо сам запускает тайм-аут, либо может подождать в надежде, что тренер другой команды закажет тайм-аут, чтобы не расходовать свой.

Целевая аудитория. Проведем примерную оценку по аналогии конверсии футбольных игроков в футбольные менеджеры. Так, в 2001 г. по данным FIFA в футбол играло 250 млн человек. Предположим, что с тех пор это количество учетверилось и составило миллиард. Максимальное количество скачиваний футбольного менеджера на Android 10 млн установок. Таким образом, можно допустить, что конверсия людей, играющих в футбол в приложениях Free2play футбольного менеджера, составляет примерно 1 %. Применив эту конверсию к данным FIVB по занимающимся волейболом людям, можно предположить, что порядка 5 млн человек, интересующихся волейболом, могут заинтересоваться спортивным менеджером в этом сеттинге.

ЗАДАНИЕ ДЛЯ АНАЛИТИЧЕСКОГО ОБЗОРА

Необходимо выбрать две игры. Обзор первой игры – учебный, на нем корректируем понимание структуры игры. Второй обзор – контрольный. Темы для обзора представлены в табл. 2.3.

План аналитического обзора:

- классификация игры;
- жанр;
- сеттинг;
- целевая аудитория;
- список элементов игры;
- список циклов в игре;
- метагейм.

Таблица 2.3

Задания для аналитического обзора согласно варианту

Номер варианта	Темы для обзора игр
1	Игра для игрового автомата
2	Стелс-экшн
3	Квест
4	Экшн
5	Стратегии
6	Спортивные симуляторы
7	Аркада
8	Музыкальные игры
9	Головоломки
10	РПГ
11	Настольные игры
12	Бродилки-стрелялки
13	Тактические шутеры

Лабораторная работа №3

СОСТАВЛЕНИЕ КОНЦЕПЦИИ КОМПЬЮТЕРНОЙ ИГРЫ ПО ЗАДАНЫМ ТРЕБОВАНИЯМ

Рассмотрим пример составления концепт-документа компьютерной игры.

Для чего нужен концепт-документ?

Для того чтобы кратко описать суть и основные элементы игры, а потом показывать этот документ новым членам команды разработки, возможным инвесторам. Концепт-документ не требует большого количества времени на чтение, но дает более-менее полное представление о том, каким получится проект на выходе.

Файл концепт-документа является не просто текстовым документом, а содержит дополнительное оформление.

Цели этого оформления:

1. Предоставить читателям дополнительные сведения и вместе с тем персонализировать документ.

2. Более-менее подробно изложить суть игры, так чтобы возможный инвестор или новый член команды понял, с чем он имеет дело, не потратив очень много времени на чтение документа.

3. Добавить документу солидности. Здесь важно знать меру – необходимо показать, с одной стороны, что концепт сделан не «на коленке», а с другой – что мы имеем не оформление ради оформления, оно информативно и лаконично.

Отдельно отмечается такая полезная вещь, как автоматически создаваемое оглавление, содержащее ссылки на указанные в нем разделы. Оно ускоряет доступ к нужным фрагментам документа, хотя концепт должен читаться за один раз, иногда необходимо вернуться и освежить в памяти какие-то моменты.

Количество страниц с учетом титульного листа, оглавления и раздела контактов должно быть не больше 6–7 страниц. Пожалуй, такой объем вполне удовлетворителен – документ по-прежнему можно быстро прочесть, при этом он является вполне информативным.

Что включает в себя концепт-документ:

1. Краткое описание игры: какие ощущения игра должна вызывать, на что нужно сделать упор, чем/кем управляет игрок, что он делает и с какой целью/результатом, какова конечная цель игрока.

2. Жанр, Сеттинг, ЦА, платформа.

3. USP – unique selling point – ключевые особенности игры, которые отличают ее от аналогов и обеспечивают ей популярность среди игроков. Следует отметить, что в настоящее время чаще используется понятие Selling Point, т. к. ключевыми особенностями игры являются не только совершенно новые, никем ранее не придуманные механики, но и правильно выбранные сочетания особенностей других похожих игр.

4. Подробное описание игры: список основных игровых элементов (механик), принцип их работы, описание основного игрового цикла (циклов).

Примечание. Последние два пункта можно объединить, расписывая сразу все игровые элементы, также можно включить описание жанра и сеттинга игры в ее краткое описание.

Рассмотрим пример концепт-документа на основе игры «*Ryaba's Revenge*» разработчика «1C Games».

Краткое описание

Игра – стратегия в реальном времени со связанным сюжетом и элементами RPG. Игровой процесс заключается в прямом управлении «своими» объектами (персонажем и юнитами) для выполнения квестов, а также в прохождении карты, ведении боевых действий и обеспечении ресурсами. Построение дизайна уровней основано на выполнении мини-миссий и постоянно ставит задачу принятия решений о способе прохождения того или иного участка. Игрок формирует свою «армию» грамотным подбором необходимого количества юнитов в условиях ограниченности ресурсами. Основная цель игрока – прохождение игрового сюжета (прохождение всех трех уровней, получение Золотого яйца).

Целевая аудитория, платформа, сеттинг

Игра ориентирована на широкую аудиторию, не содержит ограничивающего контента, минимальный возраст игрока – восемь лет. Дополнительную привлекательность игра имеет для владельцев не самой современной конфигурации PC и людей, ищущих новые подходы в жанре RTS, а также для интересующихся русскими народными сказками. Для младшей части этой категории характерна потребность в простых играх.

Игра разрабатывается только в версии для PC.

Сеттинг: фэнтезийный вымышленный мир, наше время.

Ключевые особенности игры (USP)

1. Неповторимый дух русских сказок. Возможность «примерить на себя» известные с детства истории.

2. Выделенный юнит с особыми свойствами, которым игрок управляет отдельно – Курочка Ряба.

3. Нетрадиционный подход к использованию ресурсов. Теперь ресурсы нужны не для постройки юнитов, а только для их поддержания.

4. Множество загадок и мини-миссий, требующих тактических решений.

5. Невысокие требования к технике при отличной яркой графике и богатом звуке.

6. Violent-free. Убийство выполняется в лучших традициях детских игр – превращение в бабочек, цветки и подобное.

Игра пригодна для издания на западном рынке.

Сюжет рассчитан на 5–10 ч прохождения.

Основные элементы игры

1. Карта – неисследованный лабиринт, игрок должен с осторожностью открывать и осваивать новые области.

2. Вражеские юниты – Осы, Крысы, Коты – нападают на отряды игрока.

3. Специальные объекты – Черный кот, Медведь-шатун, Лис Прохвост – обладают особыми характеристиками, могут взаимодействовать с отрядами игрока и с вражескими юнитами. Могут сотрудничать с игроком либо могут быть побеждены игроком. В бою используют приемы, недоступные обыкновенным юнитам.

4. Специальные квесты – квесты, не относящиеся к основному сюжету, но наполняющие игру контентом (открытие ворот, нахождение золотых когтей и клюва, освобождение Полкана).

5. Строения – сооружения, которые используются для производства юнитов, хранения, производства и переработки ресурсов. Некоторые строения игрок может строить самостоятельно, используя своих юнитов и затрачивая на это определенные ресурсы.

6. Курятники – строения, которые игрок строить не может. Используются для найма юнитов – Цыплят. Курятники также используются для апгрейда Цыплят путем обучения их чарам, если в них присутствует игрок юнит – Наседка. Если Наседки нет, игрок не сможет обучать юнитов в этом Курятнике.

7. Колодцы – тип строений, которые может строить игрок. Количество воды в колодце непостоянно и возможные уровни добычи воды неравномерно распределены по местности картой «уровня вод», задаваемой при дизайне уровней.

8. Кормушки – тип строений, которые игрок строить может, располагаются (могут и должны располагаться) недалеко от курятников, используются для апгрейдов Цыплят – их превращения в Петушков и Жар-птиц.

9. Команды – используются игроком для управления отрядами. Стандартные команды, отвечающие за передвижение и бой юнитов, одинаковы для всех; некоторые юниты имеют дополнительные возможности и соответственно дополнительные команды.

10. Вода – единственно необходимый и доступный ресурс. Поиск воды является одним из видов деятельности, которым занимается игрок. Вода находится в бидоне и расходуется со скоростью, зависящей от размера войска и активности боевых действий. Для разработки этого ресурса используются юниты – Цыплята, которые носят воду из водоемов или колодцев.

Игровой цикл: игрок формирует отряд, добывает ресурсы, сражается, выполняет квесты → тратит опыт и ресурсы на улучшение боевых отрядов → повторяет свои действия.

При достижении игроком конечной цели – прохождении все трех уровней и получение Золотого яйца – игра заканчивается. Продолжать играть можно сначала или с любого сохраненного места.

Пример геймплея

Вид игры – изометрия в $\frac{3}{4}$, тайловый ландшафт с дискретным изменением высоты, плоский интерфейс пользователя, управление – полностью мышью с возможностью назначения быстрых вызовов на клавиатуре (рис. 3.1).

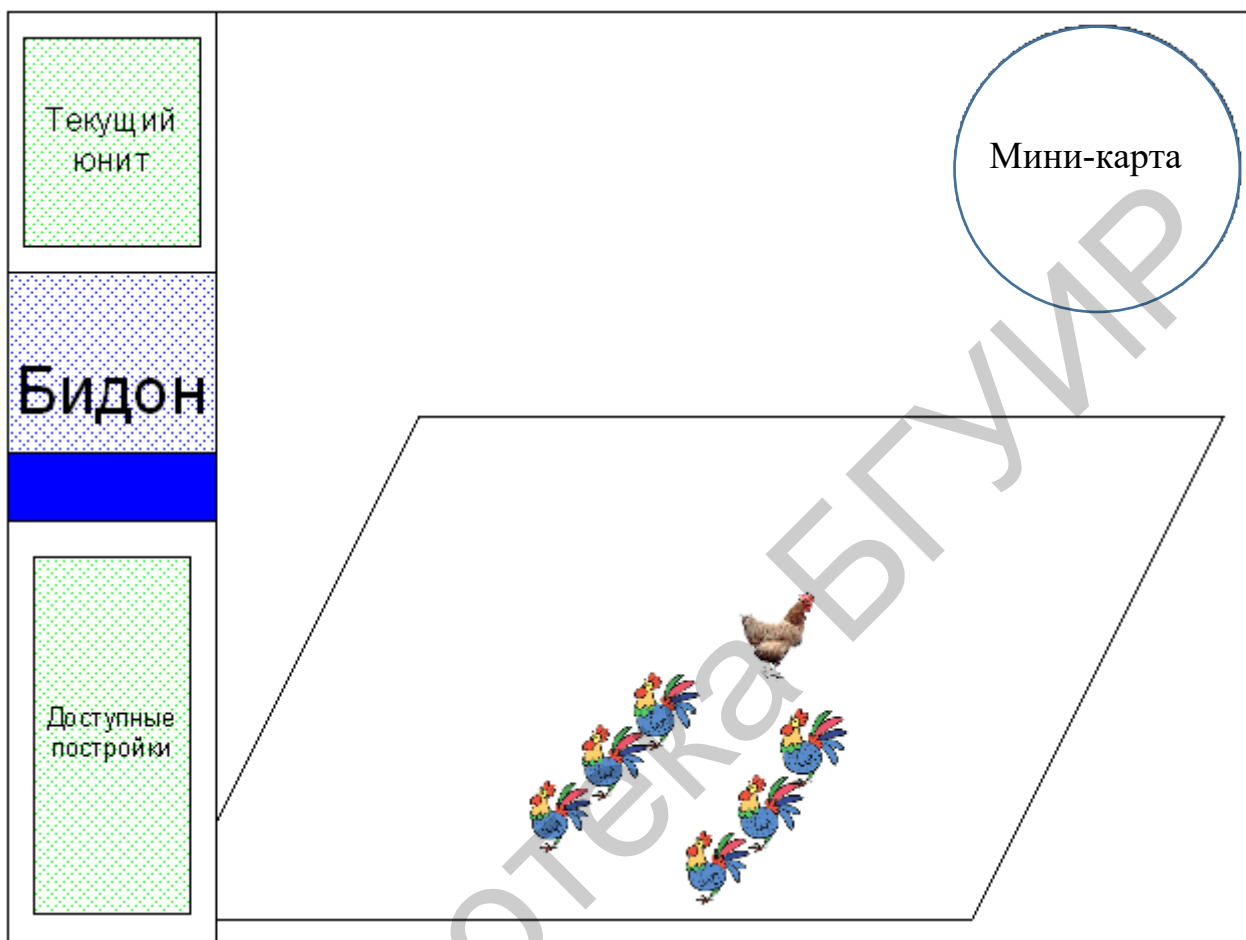


Рис. 3.1. Арт игрового приложения

Интерфейс функционально привычен и стандартен для RTS, за исключением бидона как визуального индикатора наличия самого важного ресурса – воды.

Стиль юнитов и построек мультяшно-сказочный, красочный, озвучена игра звонко, весело, понятно даже самым маленьким игрокам.

Общий стиль графики – лубочный славянский, вписывающийся в сеттинг и подчеркивающий атмосферу сказочности игры (рис. 3.2).



Рис. 3.2. Пример графики приложения

Общая цель миссии – вырастить 20 цыплят.

Начальные условия: стартовая позиция возле колодца, одна Ряба, одна Наседка.

Задание: вырастить пять цыплят. Разведать территорию вокруг хутора.

Игрок выбирает простое гнездовище (Панель «Строить» → Гнездовище простое) и строит его неподалеку от колодца (LClick на точке строительства). Выбирает Наседку (LClick на ней) и привязывает к гнездовищу (RClick на гнездовище). Наседка подходит к гнездовищу и начинает высиживать яйца. В статусной панели игрок видит истекающую полосу, символизирующую таймер откладывания яиц.

Игрок выделяет Рябу и начинает разведывать территорию вокруг хутора, направляя движение курицы (LClick на ландшафте или LClick на мини-карте). Игрок находит ручей и стог сена рядом, возвращает камеру к колодцу (LClick на мини-карте или скроллинг). Там уже выращены пять цыплят, и наседка освободилась. Они объединяются в группу (рамка, Ctrl+1) и отправляются в район ручья (LClick на мини-карте). У ручья игрок строит новое гнездовище (Панель «Строить» → Гнездовище из сена).

Задание: обеспечить семью водой.

Игрок выделяет пять цыплят (рамка), которые назначаются на добычу воды (RClick на ручье). Цыплята начинают таскать воду в гнездовище. Бидон в интерфейсе медленно наполняется.

Игрок тем временем разведывает территорию дальше с помощью Рябы. Он замечает Кота, крадущегося в направлении ручья и нового гнездовища.

Задание: уберечь цыплят от Кота.

Кот подбегает к цыплятам,носящим воду, и хватает одного. На месте цыпленка появляется желтенький цветочек. Игрок же отдает приказ Рябе атаковать (RClick на Коте), она кудахчет и клюет Кота. Тот убегает в туман войны.

Задание: высидеть больше цыплят.

Воды достаточно, и игрок строит кормушку (Панель «Строить» → Кормушка простая), после чего отправляет Наседку в гнездовище (LClick на ней, RClick на гнездовище). Наседка продолжает производить цыплят. По мере появления цыплята направляются на добычу воды к ручью (рамка, RClick на ручье).

Задание: исследовать территорию вдоль ручья.

Ряба под управлением игрока продолжает исследовать местность вдоль ручья. Вскоре находятся залежи орешков, вокруг которых летают Осы. Игрок оставляет Рябу на безопасном расстоянии в один экран и вызывает цыплят (Shift+1, LClick на ландшафте). Рябе отдается приказ распугивать врагов (команда Guard) – с кудахтаньем и размахивая крыльями она бросается на Ос. Осы разлетаются со злобным жужжанием, но в целом продолжают крутиться над залежами ресурсов. Цыплята приходят, и игрок приказывает строить новую кормушку (Панель «Строить» → Кормушка большая). Цыплята строят ее, а Ряба тем временем продолжает отгонять Ос. Но одной из Ос удастся ужалить цыпленка, он превращается в бабочку и улетает.

Задание: построить отряд Петушков и исследовать другой берег.

Кормушка готова, Осы разогнаны, и игрок начинает улучшать цыплят, посылая их поочередно в кормушку (выделение цыпленка, RClick на большой кормушке). Полученные молодые петушки умеют форсировать ручей, и игрок направляет их вместе с Рябой на другой берег. Там быстро находится большая залежь ресурсов, охраняемая отрядом Навозных жуков. Девять Петушков под контролем игрока легко их разгоняют (рамка, RClick на любом враге) одного за другим.

Задание: высидеть больше цыплят.

Игрок строит еще одну кормушку (Панель «Строить» → Кормушка большая), гнездовище (Панель «Строить» → Гнездовище простое), Наседка форсирует ручей и высиживает яйца. Вылупляется 20-й цыпленок и миссия заканчивается.

Сравнение и особенности позиционирования

Игра «Ryaba's Revenge» имеет, с одной стороны, некоторые оригинальные решения в жанре, но в то же время концепция игры использует следующие лучшие свойства выбранных образцов:

1. Warcraft – классика жанра RTS: исследование карты, управление юнитами, обеспечение ресурсами. Также использован подход жесткого сюжета, который выступает как центральная ведущая линия и дает минимум свободы в выборе целей.

2. Князь – использование славянского сеттинга.

3. Wargames – минимальный менеджмент ресурсов, акцент на планировании действий и ведении сражения.

4. Казаки – применен подход, в котором ресурсы тратятся постоянно, в зависимости от размера войска.

5. C&C, Warcraft III – выделенный персонаж, игровое взаимодействие с которым ближе к RPG, чем RTS.

Данный подбор свойств игры ориентировался на такие современные тенденции в RTS, как использование разнообразных сеттингов, «эксперименты» с ресурсами, использование RPG-элементов для юнитов.

Основные отличия от конкурентов:

1. Полное отсутствие жестокости, крови, убийств, ориентация на детскую аудиторию

2. Игра короче и проще, более насыщена роликами и внутриигровыми событиями, больший упор на повествование, а не стратегическое развитие. Игра ориентирована на максимально казуальную аудиторию.

3. Необычный сеттинг, визуально и по звуковому ряду больше похожий на картинки из детских книжек, мультфильмы.

Платформа

Разработка игры для иных платформ кроме PC Windows не планируется. Необходимые требования к компьютеру представлены в табл. 3.1.

Таблица 3.1

Требования к характеристикам ПК

Требования	Минимальные	Рекомендуемые
Процессор	Celeron 266 МГц	Pentium 3400 МГц
ОЗУ	128 МБ	256 МБ
Видеокарта	DirectX 5 совместимая, класса ATI RAGE	

Контакты

Контактные лица: xxx

Телефоны: xxx

E-mail: xxx

Адрес: xxx

Сайт: xxx

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Составить концепт компьютерной игры по заданным требованиям (табл. 3.2).

План описания концепта:

1. Видение.
2. Сеттинг. Целевая аудитория.
3. Описание того, что делает игрок.
4. Детализация того, что делает игрок, описание наиболее важных элементов и главного героя.
5. Список USP – уникальные, неповторимые элементы в игре, которые привлекут пользователя.

Составленный концепт не должен превышать по объему двух страниц формата А4.

В ходе формирования задания студенты выбрасывают кубики с нанесенными на грани изображениями. Всего фиксируется девять изображений. Студент включает все эти изображения в составляемый концепт в любом качестве (они могут быть использованы в качестве персонажей, фона, пейзажа).

Таблица 3.2

Темы для создания концепта игры

Номер варианта	Темы для обзора игр
1	Игра для игрового автомата
2	Стэлз-экшн
3	Квест
4	Экшн
5	Стратегии
6	Спортивные симуляторы
7	Аркада
8	Музыкальные игры
9	Головоломки
10	РПГ
11	Настольные игры
12	Бродилки-стрелялки
13	Тактические шутеры

Лабораторная работа №4

ЗНАКОМСТВО СО SCRATCH. ПРОТОТИПИРОВАНИЕ КОНЦЕПЦИИ КОМПЬЮТЕРНОЙ ИГРЫ

Краткие сведения для работы в приложении Scratch

Интерфейс Scratch и основы работы в нем.

После того как программа запустилась, появляется окно, в котором можно выделить три части (три столбца). В свою очередь каждый столбец состоит как бы из двух ячеек: верхней и нижней. Все ячейки разные и каждая из них предназначена для своей конкретной цели.

Верхняя ячейка в первом столбце содержит восемь кнопок (рис. 4.1), которые называются *движение*, *контроль*, *внешность*, *сенсоры*, *звук*, *операторы*, *перо*, *переменные*. При включении одной кнопки все остальные выключаются. Включенная кнопка вся окрашивается в соответствующий ей цвет. При этом в нижней ячейке первого столбца появляются команды, связанные с включенной кнопкой. У всех кнопок разные привязанные к ним команды. Попробуйте нажимать кнопки и посмотрите, как меняются команды в нижней ячейке.

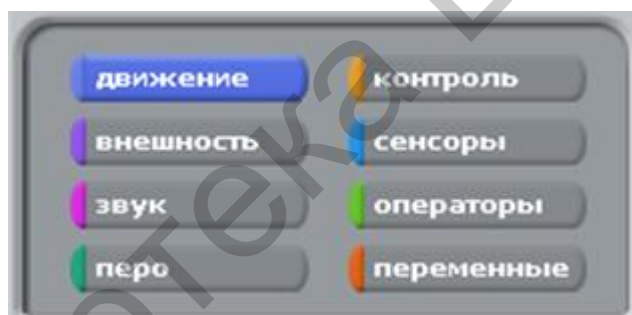


Рис. 4.1. Кнопки

На верхней ячейке второго столбца показаны свойства объекта (рис. 4.2), которым мы будем управлять (писать программы для него). Сейчас – это кот. Его имя написано в поле – Спрайт1.

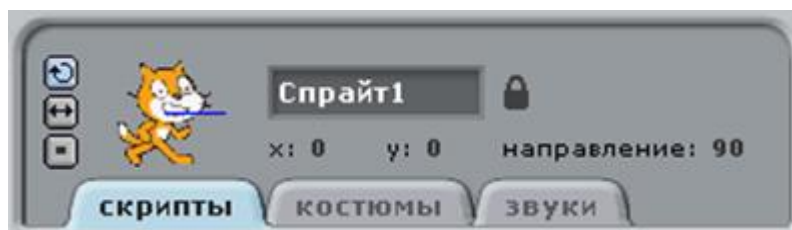


Рис. 4.2. Свойства объекта

Ниже поля имени отображены три свойства объекта – это его положение (координаты x и y) и направление. Кот смотрит направо, поэтому его направление равно 90.

Внизу имеются три кнопки-вкладки – скрипты, костюмы и звуки. От того, какая из них нажата, зависит ячейка внизу. Если нажата кнопка скрипты, то нижняя ячейка второго столбца покажет программы (скрипты) для объекта, который отображен в верхней ячейке. Кнопки костюмы и звуки позволяют настраивать и менять соответственно внешний вид объекта и издаваемые им звуки.

В третьем столбце видно поле с изображением кота. Данное поле – это холст. Многие действия, которые задает программист на вкладке скрипты, объект выполняет именно на холсте. Так, если запрограммировать ходьбу кота, то он будет перемещаться по холсту.

Если зажать левую кнопку мыши над котом, а затем, не отпуская ее, перемещать мышь, то кот будет перемещаться. Таким образом, можно менять его положение на холсте.

Кнопки, представленные на рис. 4.3, предназначены для выполнения программы и остановки.



Рис. 4.3. Кнопки пуска и остановки программы

Нижняя правая ячейка окна Scratch предназначена для работы с объектами текущего проекта. Здесь можно добавлять новые объекты и переключаться между теми, которые у нас уже имеются.

Управление несколькими объектами

1. *Координаты.* Когда окно Scratch только открылось, видно, что кот стоит в центре холста. Его координаты равны $x:0$ и $y:0$. В Scratch начало системы координат – это центр холста. От центра вправо значение x увеличивается, влево – уменьшается (становится минусовым, отрицательным). Аналогично с осью y . По направлению вверх идут положительные значения, вниз – отрицательные.

2. *Сцена.* Чтобы было легче ориентироваться в координатах, можно проделать кое-какие манипуляции с холстом, наложив на него картинку с системой координат.

Чтобы поменять фон холста, на вкладке Фоны есть кнопка *Импорт*. Если ее нажать, появляется возможность добавить картинку, которая будет служить фоном холста. На картинке *xy-grid* изображены оси координат.

3. *Новые объекты.* Для того чтобы, например, на сцене помимо кота бегала еще какая-нибудь живность, следует нажать вторую кнопку в ряду кнопок под холстом (рис. 4.4).



Рис. 4.4. Ряд кнопок, расположенных под холстом

После чего открывается диалоговое окно, где из папок Animals (Животные), Fantasy (Фантазия), Letters (Буквы), People (Люди), Things (Вещи) и Transportation (Транспорт) можно выбрать любой объект. Объект добавляется в центр холста.

Последовательное и одновременное выполнение

1. *Одновременное выполнение скриптов (программ).* В Scratch можно сделать так, что два (или больше) скрипта одновременно будут выполнять один объект. Допустим, кот будет шагать и при этом менять свои размеры, форму и другие свойства. При этом ходьба и изменение свойств друг с другом никак не будут связаны. Сделать это можно, например, разместив два разных скрипта в ячейке скрипты для кота (рис. 4.5).

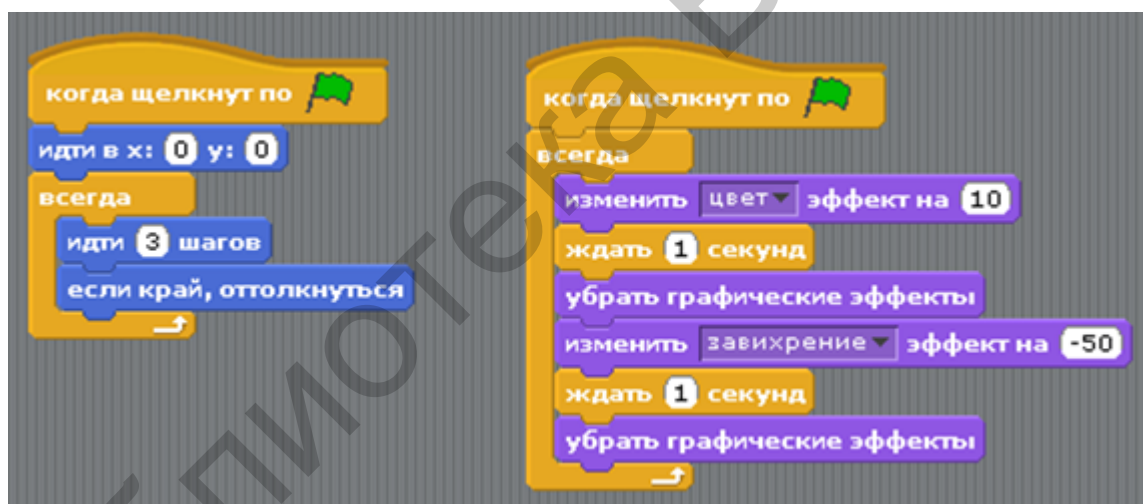


Рис. 4.5. Одновременное выполнение скриптов

При щелчке на кнопке запуска начнут работать оба скрипта сразу. Первый будет заставлять кота ходить слева направо. Второй скрипт будет менять внешний вид кота. Фиолетовые команды связаны с кнопкой внешность. Во втором скрипте выше используется команда «изменить ... эффект на ...». После слова «изменить» в раскрывающемся списке можно выбрать понравившийся нам эффект, а в поле с числом прописать, на сколько единиц его изменять. Команда «убрать графические эффекты» возвращает объект к его исходному внешнему виду. Итак, второй скрипт в цикле «всегда» выполняет следующие действия: изменяет цвет объекта → оставляет объект в таком состоянии на 1 с → возвращает объект к прежнему цвету → искривляет объект с помощью

завихрения → оставляет в таком состоянии на 1 с → возвращает к исходной форме.

2. *Последовательное выполнение скриптов (программ).* При последовательном выполнении скриптов сначала все действия должен совершать один объект, затем второй и т. д. Как же это можно сделать в Scratch?

Например, можно просто использовать команду «ждать» в начале скрипта второго объекта. Но на самом деле это не будет последовательным выполнением скриптов, т. к. оба они начнут работать одновременно, просто у второго объекта сначала будет долго работать команда «ждать». Хотя на сцене можно таким образом получить желаемый эффект: сначала действия будет совершать один объект, а через некоторое время – другой.

Рассмотрим более грамотный способ организации последовательного выполнения скриптов. Когда один объект завершает выполнение ряда своих команд, он должен подать какой-нибудь сигнал-сообщение второму объекту, а тот, в свою очередь, должен его принять. В Scratch для этого есть две специальные команды, связанные с кнопкой *контроль*: «передать ...» и «когда я получу ...». Команда «передать ...» дается объекту, который уже закончил все или некоторые свои действия, а «когда я получу ...» применяется к объекту, который начинает работать вторым. Вместо трех точек в этих командах вписывается сообщение, которое мы сами назначаем.

Организуем для двух объектов последовательное выполнение скриптов. Допустим, по нашему сценарию планируется такая анимация: например, кот сначала ходит туда-сюда, после чего исчезает, и по экрану начинает метаться сыр.

Первый скрипт кота представлен на рисунке 4.6.



Рис. 4.6. Скрипт первого кота

В команде «передать ...» надо вставить сообщение. Для этого открывается специальное окно, куда можно его вписать. Команда «спрятаться» используется для того, чтобы скрыть кота, после того как он пошлет сообщение. Команда «показаться» в начале скрипта требуется, чтобы снова отобразить

объект, если он был скрыт до этого. Понятно, что при первом выполнении скрипта она не играет никакой роли.

Теперь посмотрим, что будет делать второй объект. Он должен получить сообщение (рис. 4.7), после чего совершить какие-нибудь действия.



Рис. 4.7. Скрипт второго кота

Таким образом, второй скрипт не начнет выполняться, пока первый не передаст ему сообщение «первый вызывает второго». Обратите внимание, что команда «*скрыться*» первого объекта находится ниже, чем «*передать ...*». Это значит, что у нас не совсем последовательное выполнение скриптов. Ведь первый еще не завершается, когда начинается второй. В принципе, чаще именно так и бывает.

3. *Изменение размеров объектов.* Можно заметить, что холст слишком маленький, а добавляемые на него объекты зачастую слишком крупные. В Scratch можно менять размер объектов вручную или программно.

В первом случае используются две последние кнопки в ряду из четырех кнопок вверху (рис. 4.8).



Рис. 4.8. Кнопки последнего ряда

Здесь первая кнопка позволяет дублировать (копировать) объект. При этом копируются и все его скрипты. Вторая кнопка предназначена для удаления объекта. Может случиться, что добавленный объект оказался лишним. Его можно будет удалить с помощью этой кнопки. Третья кнопка отвечает за увеличение объекта, а четвертая – за его уменьшение.

После того как любая из этих кнопок выбрана, указатель мыши меняет свой вид. Далее следует щелкнуть на объекте, который мы планировали изменять. После окончания изменений следует щелкнуть кнопкой мыши в пустом месте холста, чтобы сбросить выбранный инструмент.

Другой способ изменить размер объектов – использование специальных команд в скрипте (программе). Это значит, что размер объекта можно изменять с помощью программы. Эти две команды представлены на рис. 4.9.

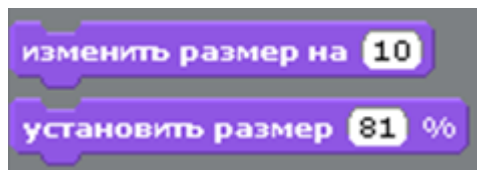


Рис. 4.9. Команды для изменения размера

Команда «изменить размер на ...» увеличивает или уменьшает объект на указанное количество точек. Положительные числа в поле будут увеличивать объект, а отрицательные (с минусом) – уменьшать его.

Команда «установить размер ... %» вычисляет размер объекта, относительно его оригинального (самого первоначального) размера. 100 % – это и есть оригинальный размер. Если прописать в поле 50, то объект уменьшится в два раза, а если 25, то в четыре. Например, чтобы увеличить героя в два раза, требуется задать 200 %.

Интерактивность, условия и переменные

1. *Интерактивность.* Под интерактивностью будем понимать возможность взаимодействия между объектами, принадлежащими разным средам. Например, если в написанной нами программе задано, что кот из Scratch догоняет мышь из Scratch, а она реагирует на него и убегает, то это не будем считать интерактивностью. И кот и мышь – объекты одной среды. А вот если кот из Scratch реагирует на действия реального человека (например, нажатие пользователем определенной клавиши), то это уже интерактивность, т. к. объекты принадлежат разным «средам обитания».

Пусть требуется сделать так, чтобы человек мог управлять поведением кота на холсте Scratch. Если написать такой сценарий (рис. 4.10), то после его запуска объект будет преследовать указатель, если удерживать зажатой левую кнопку мыши.

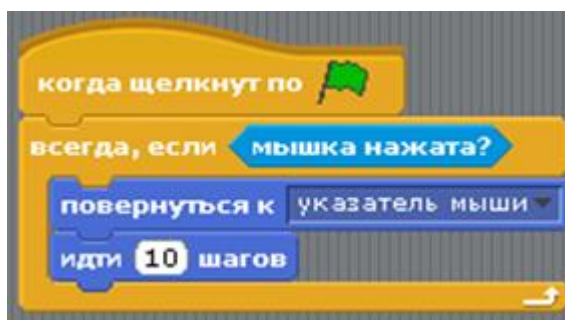


Рис. 4.10. Сценарий

Команда «мышка нажата?» находится в разделе сенсоры, где сгруппированы команды, назначение которых – проверять наличие заданных изменений в окружающей среде объекта. Например, команда «мышка по x» отслеживает, где находится курсор мыши по оси x.

Управляющая конструкция «*всегда, если ...*» похожа на цикл «*всегда*», за исключением того, что эта конструкция сочетает в себе цикл с условием. Такой цикл выполняется лишь тогда, когда условие истинно. В данном случае, когда кнопка мыши нажата.

2. *Переменные и условный оператор.* Усложним программу следующим образом. Пусть объекты перемещаются не вместе, а врозь: сначала один бежит за курсором, потом – другой. Переключение «активности» между объектами пусть происходит с помощью клавиши **пробел**. Как организовать такую программу?

Ответить на этот вопрос будет очень сложно, если не знать об очень важной составляющей любого программирования – о **переменных**. Переменные представляют собой ячейки, в которых хранятся какие-либо данные (число, строка и др.). Причем содержимое ячеек может меняться в процессе выполнения программы. В какой-то момент можно проверить, что там находится и в зависимости от этого выполнять определенные команды или нет.

Создадим ячейку-переменную под именем **идуший**. Для этого нужно нажать кнопку **переменные** и затем в ячейке команд Scratch нажать **Создать переменную**; в открывшемся диалоговом окне ввести ее имя и нажать **ОК**. После чего у нас появится сама переменная и несколько команд для ее изменения.

Нам нужна эта переменная, чтобы в зависимости от ее значения двигался первый объект или второй. Пусть если **идуший = 0**, то двигается первый объект, а если переменная равна единице, то – второй. Изменение значения переменной будет происходить при нажатии клавиши **пробел**. Поскольку данная часть программы относится к обоим объектам, то уместно расположить скрипт в объекте **Сцена** (хотя не обязательно), переключившись на него. Этот сценарий представлен на рис. 4.11.



Рис. 4.11. Сценарий

Вместо пробела можно выбрать другую клавишу в раскрывающемся списке первой команды сценария. Далее идет сложная инструкция «если – или». Она работает так: если условие в части «если» истинно, то выполняются вложенные команды после «если», иначе – будут выполняться вложенные команды после «или». Обе «ветки» никогда вместе не выполняются, они взаимно исключают друг друга.

Теперь рассмотрим условие при инструкции «если». Зеленые команды относятся к **операторам**. Операторы – это такие команды, которые производят какие-либо действия над данными. Например, выполняют математические операции или сравнивают числа между собой. В данном случае выбран оператор сравнения на равенство. Он проверяет равно ли выражение слева от знака «=» выражению справа. Если да, то будет работать код после «если», если нет, то код после «или». В нашем случае сравниваются значение переменной «идуший» и 0.

Что же все-таки делает этот скрипт? Если значение переменной равно 0, то оно меняется на 1. Если же нет, то меняется на 0. Получается, что нажатие клавиши **пробел** попеременно меняет значение переменной «идуший». Однако при выполнении программы мы не увидим никаких изменений. Действительно, ведь объекты еще «не знают», к какому значению переменной привязаны их действия. Составьте вышеприведенный скрипт и переключитесь на первый объект.

Требуется организовать проверку значения переменной. Для этой цели подойдет простая инструкция «если». На рис. 4.12 показаны сценарии, которые получаются для первого и второго объектов.

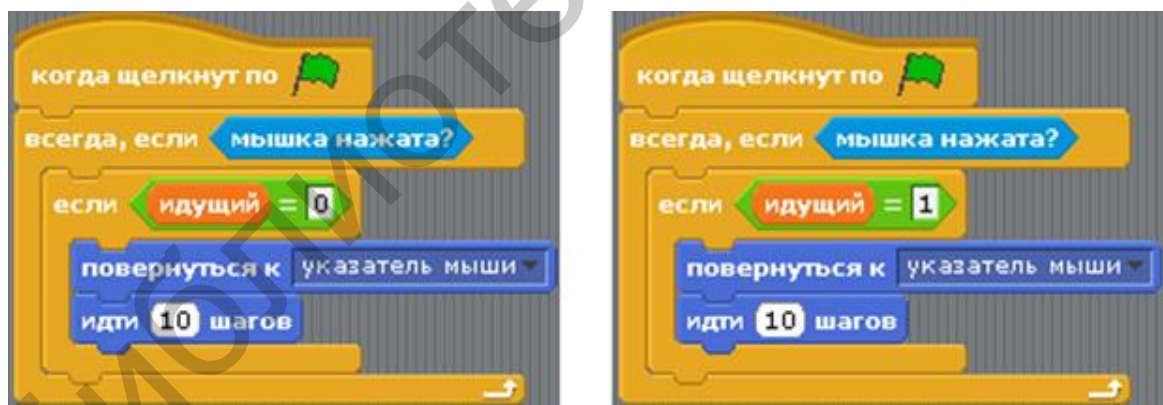


Рис. 4.12. Сценарии для первого и второго объектов

Разница между ними только в том, с чем сравнивается значение переменной: с нулем или единицей. А как мы знаем, ее значение меняется при нажатии пробела. В результате, когда программа запущена, мы можем менять «активный» объект.

Случайные числа

Предположим, что требуется в Scratch реализовать следующий сценарий (рис. 4.13). Под водой моллюск пытается поймать пищу, которая от него постоянно отпрыгивает в новое случайное место. Через какое-то время моллюску все-таки удастся поймать и съесть еду.



Рис. 4.13. Сценарий для игры

Итак, моллюск всегда должен двигаться за едой. Причем через определенный ряд действий это должно прекратиться. Программа для моллюска может быть такой, как показано на рис. 4.14.

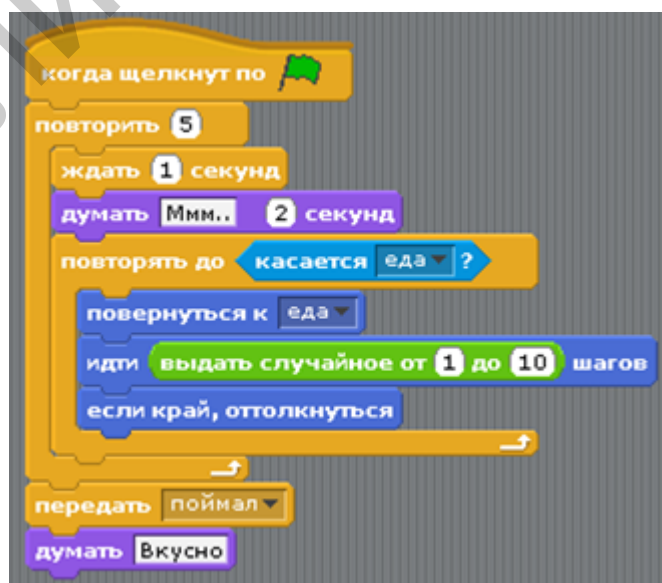


Рис. 4.14. Программа для моллюска

Проследим логику работы этого сценария. Сначала моллюск стоит и думает. После чего он поворачивается к еде и идет к ней до тех пор, пока не коснется ее. Все эти действия повторяются пять раз. Обратите внимание, в команду «идти ... шагов» вставлена команда «выдать случайное число от ... до ...». Вторая команда выдает случайное для нас число в указанном диапазоне, т. е. любое число от первого числа в команде до второго включительно. В данном случае эта команда влияет на то, с какой скоростью двигается моллюск.

После того как моллюск совершит несколько вышеописанных циклов, он передаст в окружающую среду сообщение «поймал». В результате у него должна появиться соответствующая ситуации мысль.

Если сейчас запустить программу, то моллюск подойдет к еде, пять раз подумает «Ммм...» и один раз «Вкусно».

Теперь подумаем, что должна делать еда (рис. 4.15).

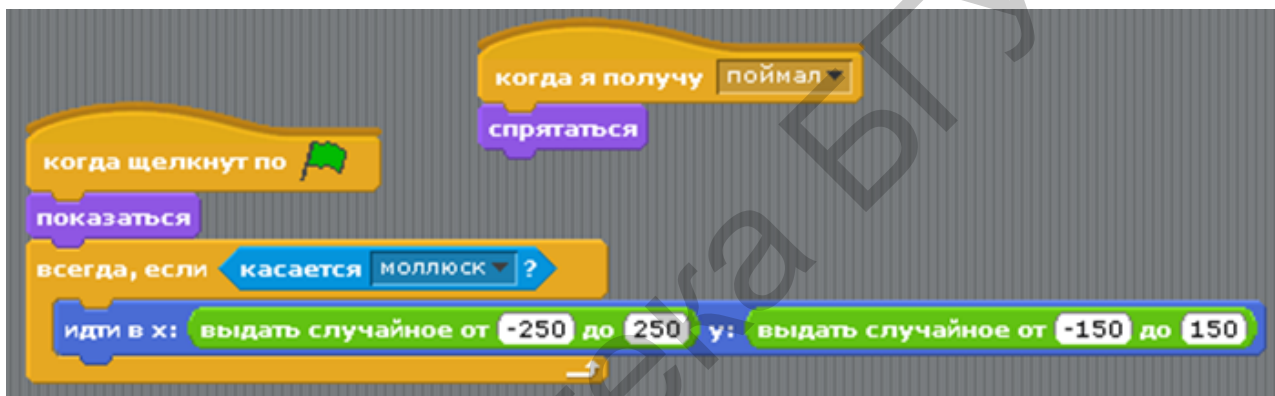


Рис. 4.15. Сценарий для еды

Она всегда должна оказываться в неожиданном для моллюска месте, если тот касается ее. «Случайное место» определяется двумя командами «идти в x: ... y: ...» и «выдать случайное число от ... до ...». То есть, после того как координаты x и y определяются случайным образом, объект перемещается в заданную точку.

Второй сценарий еды очень прост. Если в «пространстве» появляется сообщение «поймал», то еда должна его получить и спрятаться.

Анимацию можно сделать интересней, если придать сцене больший динамизм. Пусть картинка сцены слегка видоизменяется, причем тоже случайным образом. Рассмотрим следующий сценарий (рис. 4.16).

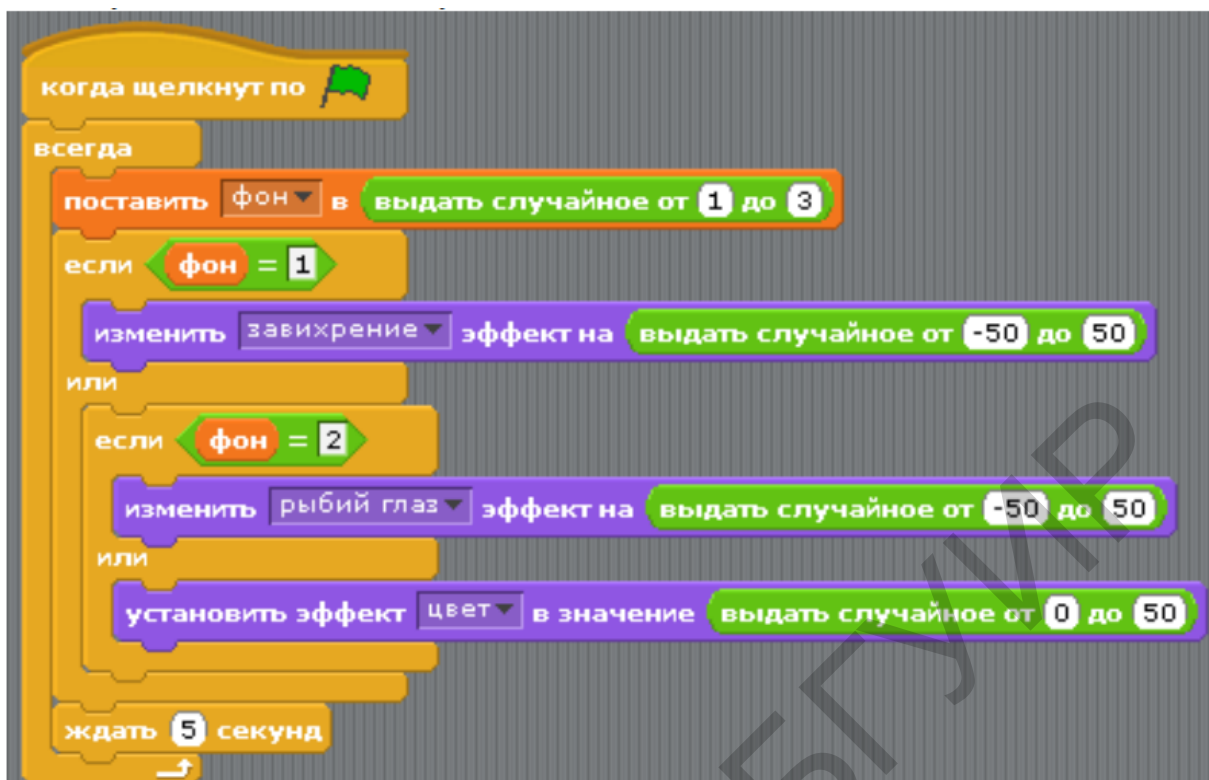


Рис. 4.16. Сценарий для сцены

В нем присутствует переменная «фон», значение которой определяется случайно, и это значение может быть только числами 1, 2 или 3. Если фон равен 1, то на картинке появятся завихрения случайной силы, если фон равен 2, то появится эффект «рыбий глаз». Во всех остальных случаях будет меняться цвет фона. Значение переменной будет обновляться через каждые 5 с.

Диалог с программой

Языки программирования по сути выполняют те или иные операции над данными (числами, строками, списками и др.). Эти данные в программу может «заложить» программист или же они будут поступать от пользователя в процессе выполнения программы. Например, если программа выполняет умножение числа на само себя, то это число может быть получено от пользователя. Чтобы пользователь ввел число, должна выполняться какая-то встроенная команда, предоставляющая интерфейс для ввода данных.

В Scratch команду, позволяющую пользователю ввести данные, можно найти в разделе *Сенсоры*. Она называется «спросить ... и ждать». Вместо фразы «Как ваше имя?» можно вписать другую фразу, а также поставить туда переменную или что-то другое. Когда команда «спросить ... и ждать» выполняется, то внизу холста появляется поле для ввода. Пользователь должен туда ввести данные и нажать **Enter** или галку в конце поля.

После того как команда «спросить ... и ждать» выполнена, текстовое поле исчезает с холста. Куда же попадают данные, которые мы в него вводили?

Они оказываются в переменной «ответ». Это встроенная переменная (ее не нужно создавать), и находится она в сенсорах.

Например, составим сценарий, при выполнении которого кот спрашивает у нас имя, а затем приветствует нас по имени. Для того чтобы получить фразу, например «Привет, Саша», следует соединить «Привет,» и то, что ввел пользователь. Это можно сделать с помощью команды «слить», находящейся в операторах (рис. 4.17).

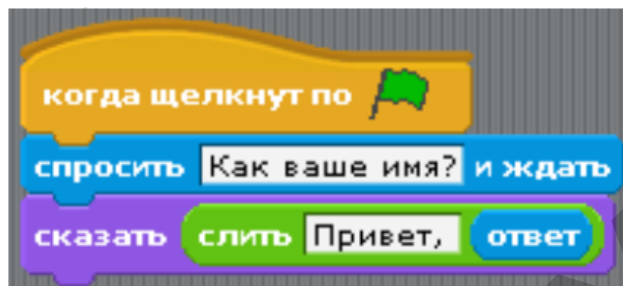


Рис. 4.17. Использование команды «слить»

Если необходимо повторять вопрос до тех пор, пока пользователь не даст правильный ответ, то удобно использовать конструкцию «повторять до ...». Условием, когда эта конструкция должна прекратить выполняться, будет правдивость результата выполнения операции. Например, пусть, после того как кот поздоровается, он спросит, сколько будет 2×2 ? Пусть он спрашивает это, пока пользователь не даст правильный ответ, равный 4.

Создайте такой сценарий (рис. 4.18).

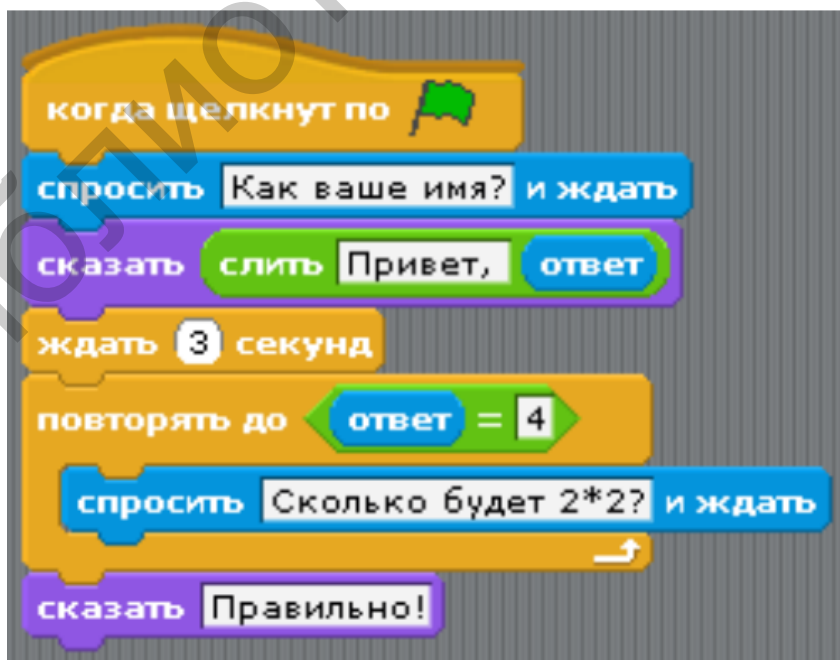


Рис. 4.18. Сценарий для ввода имени

Если теперь его запустить на выполнение, то произойдет следующее:

1. Кот спросит имя.
2. Поздоровается. Чтобы приветствие сразу не исчезло, ждет 3 с.
3. Будет спрашивать «Сколько будет $2*2$?» до тех пор, пока пользователь не введет правильный ответ.
4. Когда пользователь введет цифру 4, кот скажет «Правильно!».

Создание объектов и костюмов

Обычно Scratch поставляется с большой коллекцией готовых спрайтов и фонов. Новички в Scratch охотно ими пользуются. Но что делать, если вы придумали какой-то сценарий или программу, для которой требуются объекты, которых нет? Их можно нарисовать самим прямо в среде Scratch. Здесь есть встроенный редактор, позволяющий создавать спрайты. Чтобы его запустить, нужно нажать кнопку **Рисовать новый объект** в ячейке, где отображаются объекты программы (нижняя правая ячейка Scratch).

1. *Описание графического редактора.* Здесь можно делать следующее (начнем сверху):

- изменять размер объекта, поворачивать его по часовой и против часовой стрелки, переворачивать объект по горизонтали и вертикали;
- импортировать готовый объект, чтобы изменить его;
- полностью очищать рабочую область;
- отменять действия, а также применять их снова;
- использовать для рисования кисть и геометрические примитивы (эллипс, прямоугольник, линия); при использовании кисти и линии можно выбрать их толщину, эллипсы и прямоугольники могут представлять собой контуры или заполненные области;
- удалять элементы ластиком;
- выбирать цвет и заливать им области; заливка может быть сплошной или градиентной;
- работать с текстом;
- выделять области изображения для перемещения или дублирования;
- можно задавать цвет фона; это важно, например, при градиентной заливке;
- можно менять масштаб изображения.

2. *Создание объектов.* Создадим в Scratch следующую программу. На холсте находится смайлик, пульт и указатель. В зависимости от того, где указатель располагается на пульте, смайлик меняет выражение своего «лица». Положение указателя пусть определяется с помощью цвета, а не координат.

Для реализации такой программы нам понадобится создать примерно такие объекты, как представлены на рис. 4.19.

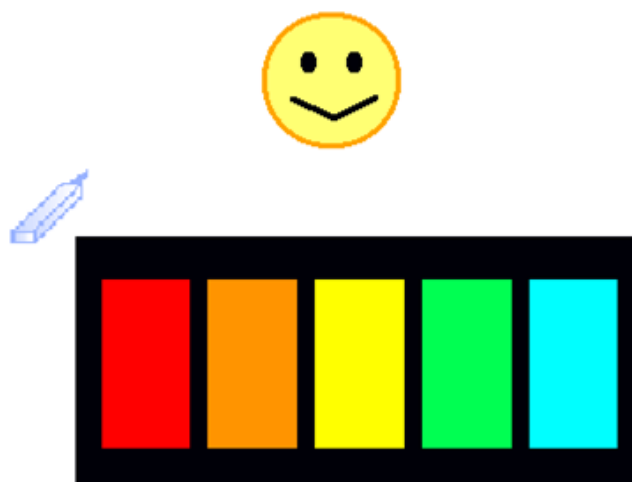


Рис. 4.19. Пример прорисовки смайлика

3. *Создание костюмов.* Для изменения объектов в Scratch используется понятие *костюмов*. Каждый объект имеет хотя бы один костюм. В этом можно убедиться, если перейти на вкладку **костюмы** в среднем столбце окна Scratch. Чтобы добавить новый костюм, надо нажать кнопку **Рисовать**, **Импорт** или **Камера**. Однако чаще всего придется изменять уже готовый костюм. В этом случае копируют уже существующий костюм, а затем редактируют его (рис. 4.20).

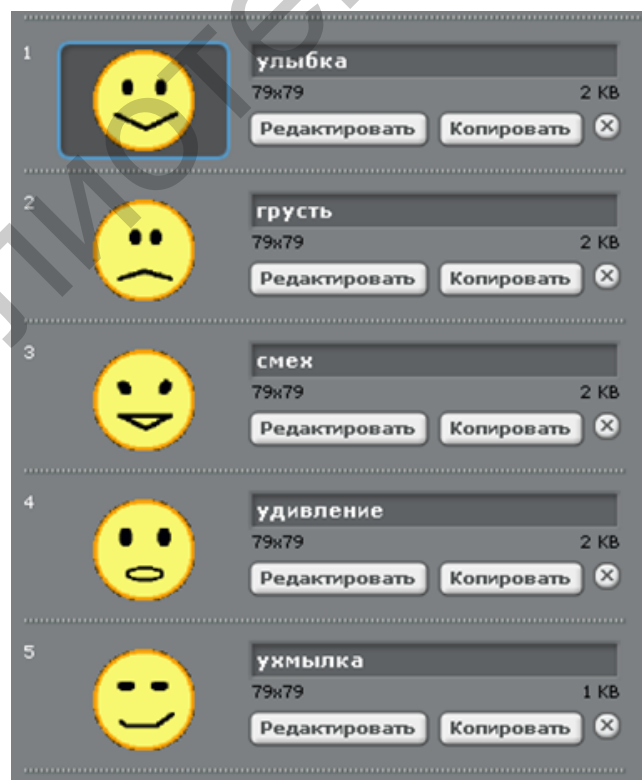


Рис. 4.20. Пять разных костюмов для смайлика

4. *Создание программы.* Как же связать положение указателя с костюмом совершенно другого объекта (смайла)? Можно использовать сообщения. Когда указатель касается заданного цвета, то он передает соответствующее этому цвету сообщение. Смайл в свою очередь «ловит» сообщение и применяет тот костюм, который ему соответствует. Вот какие должны быть сценарии для указателя и смайлика (рис. 4.21).

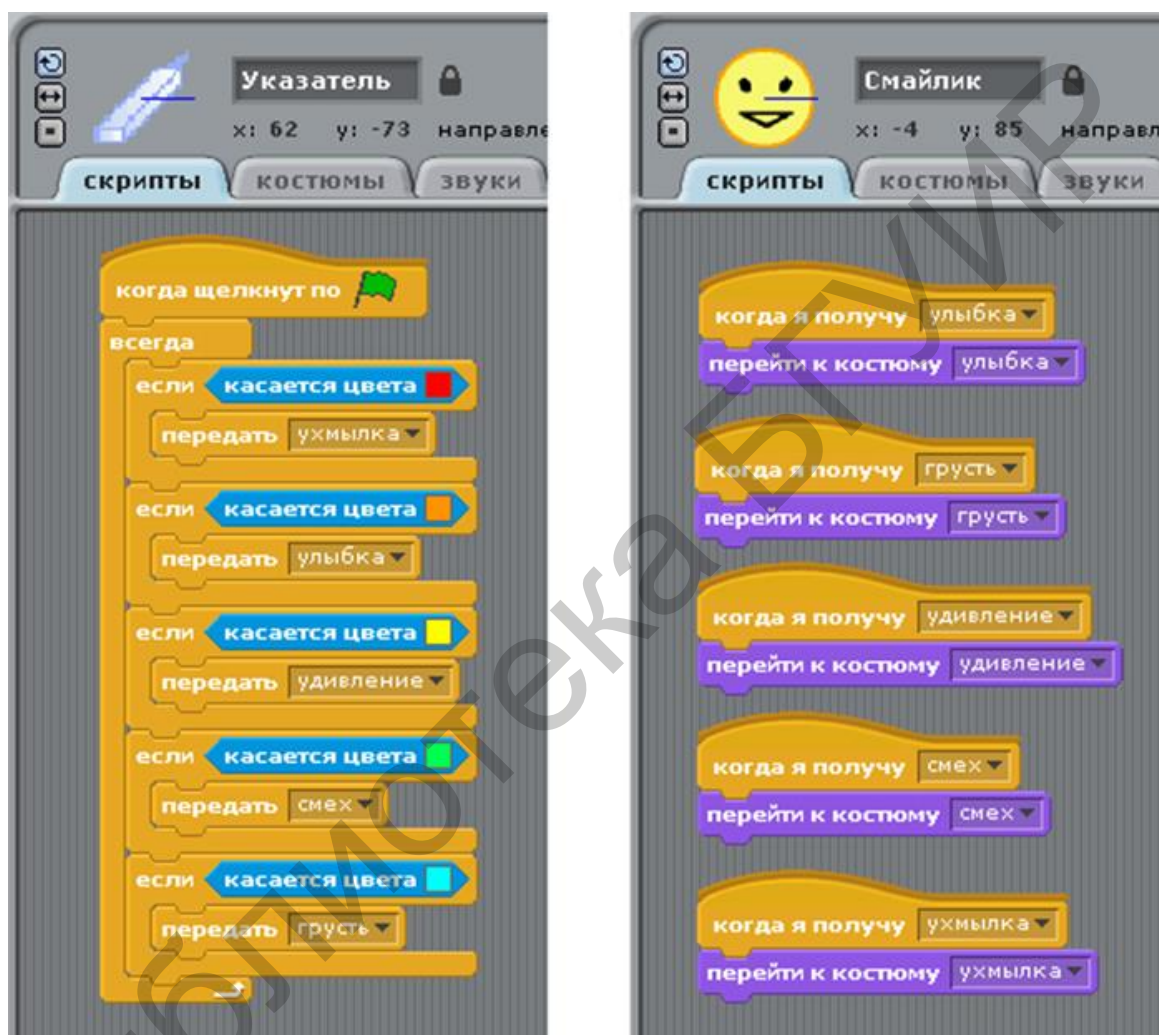


Рис. 4.21. Сценарии для указателя и смайлика

Использование библиотеки объектов

Когда мы добавляем кем-то ранее подготовленный объект из библиотеки Scratch (кнопка **Выбрать новый объект из файла**), то можно сказать, что мы импортируем его в наш проект. Однако мы с таким же успехом можем делать обратное: из рабочего проекта Scratch добавлять объект в общую библиотеку. Это можно назвать экспортом объектов. Потом наш объект можно использовать в других проектах наряду с другими стандартными (существующими до этого) объектами.

Если к объекту был «привязан» программный код, то они экспортируются и импортируются совместно.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

На основе описанного концепта продумать часть игры для прототипирования (табл. 4.1). Запрограммировать в среде Scratch выделенную часть.

Таблица 4.1

Темы для создания прототипа игрового приложения

Номер варианта	Тема анимации
1	Полет космического корабля
2	Атмосфера на ранчо
3	Плавающая рыба в воде
4	Игра на гитаре
5	Хоровод птиц
6	Движение автомобилей на перекрестке
7	Разговор между двумя людьми
8	Прыжки на скакалке
9	Танцующая пара
10	Бег собаки за брошенной палкой
11	Рост цветка с момента посадки
12	Светомузыка
13	Езда на коньках
14	Трюки на мотоцикле
15	Покупка товаров в магазине

Список использованных источников

1. Финни, К. 3D-игры. Все о разработке + приложение / К. Финни. – М. : Лаборатория знаний, 2014.
2. Роллингз, Э. Проектирование и архитектура игр / Э. Роллингз, Д. Моррис. – М. : Вильямс, 2006.
3. Zackariasson, P. The Video Game Industry: Formation, Present State, and Future (Routledge Studies in Innovation, Organization and Technology) / P. Zackariasson, T. Wilson. – London : Routledge, 2012.
4. Rogers, S. Level Up! The Guide to Great Video Game Design / S. Rogers. – NY : Wiley, 2014.
5. Brathwaite, B. Breaking Into the Game Industry: Advice for a Successful Career from Those Who Have Done It / B. Brathwaite, I. Schreiber. – Cengage Learning PTR, 2011.
6. Kent, S. The Ultimate History of Video Games: From Pong to Pokemon –The Story Behind the Craze That Touched Our Lives and Changed the World / S. Kent. – NY : Three Rivers Press, 2001.
7. Schreiber, I. Challenges for game Designers / I. Schreiber. – Cengage Learning PTR, 2011.

Глоссарий

Игровой цикл (от англ. gameplay loop, также англ. core gameplay loop) – принцип, согласно которому геймдизайнеры задают главный элемент игровой механики, который определяет фундаментальный опыт игрока. Один игровой цикл представляет собой действие игрока, результат этого действия в игровом мире, реакцию игрока на результат и запрос игры на повторение нового действия.

Интеллект-карта – визуальное представление процесса или структуры в виде дерева ассоциаций, расходящихся от центрального понятия.

Концепт-документ (концепт) – это краткое и емкое описание концепции (идеи) игры, т. е. максимально сжатый документ, в котором рассказывается о том, какой будет игра, чем она будет интересна и как она должна выглядеть после разработки.

Метагейм (от греч. meta – «за пределами», «после» и англ. game – «игра»), или метагейминг, метаигра, – использование в игре того, что относится к игрокам, а не к их персонажам.

Морфологический анализ – метод систематизации перебора вариантов всех теоретически возможных решений, основанный на анализе структуры объекта.

Сеттинг – определение принадлежности игры к какой-то сюжетной теме или к определенному виртуальному миру.

Спрайт (от англ. sprite – «фея»; «эльф») – графический объект в компьютерной графике.

Стелс (от англ. stealth – «невидимка», «скрытность») – жанр компьютерных игр, в которых игрок должен избегать обнаружения игрового персонажа противниками или скрытно устранять их, не привлекая к себе внимания.

Шутер от первого лица (от англ. First-person Shooter, FPS) – жанр компьютерных игр, в которых игровой процесс основывается на сражениях с использованием огнестрельного или любого другого оружия с видом от первого лица таким образом, чтобы игрок воспринимал происходящее глазами протагониста.

Экшн (от англ. action – «действие») – жанр компьютерных игр, в котором основное внимание уделяется перестрелкам, дракам, погоням и т. д.

Brainstorm (мозговой штурм) – оперативный метод решения проблемы на основе стимулирования творческой активности, при котором участникам обсуждения предлагают высказывать как можно большее количество вариантов решения, в том числе самых фантастичных.

Mindmap – это диаграмма связей или интеллект-карта.

Библиотека БГУИР

Содержание

Лабораторная работа №1	3
Часть 1	3
Часть 2	12
Лабораторная работа №2	20
Лабораторная работа №3	32
Лабораторная работа №4	40
Список использованных источников	56
Глоссарий	57

Библиотека БГУИР

Учебное издание

Кукин Дмитрий Петрович
Рак Татьяна Александровна
Шатилова Ольга Олеговна

**ИНДУСТРИЯ КОМПЬЮТЕРНЫХ ИГР.
ЛАБОРАТОРНЫЙ ПРАКТИКУМ**

ПОСОБИЕ

Редактор *Е. И. Костина*
Корректор *Е. Н. Батурчик*

Компьютерная правка, оригинал-макет *В. М. Задоя*

Подписано в печать 29.04.2019. Формат 60х84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 4,77. Уч.-изд. л. 4,0. Тираж 50 экз. Заказ 397.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
Ул. П. Бровки, 6, 220013, г. Минск