**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования «Калининградский государственный**

**технический университет»**

Кафедра систем управления и вычислительной техники

Допущен к защите

Декан факультета автоматизации

производства и управления

к.т.н., доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Калинин

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2018 г.

Заведующий кафедрой СУ и ВТ,

к.т.н., доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.А. Петрикин

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2018 г.

**«Автоматизированная система организации производственного процесса в студии создания графического контента»**

Выпускная квалификационная работа

09.03.01 Информатика и вычислительная техника

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

ВКР.09.03.01.ОЧ.2018.723ОЧ.17.ПЗ

Руководитель

доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Г. Пожидаев

(подпись, дата)

Работу выполнил

студент 14-ВТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Саркисян Э.В.

(подпись, дата)

Норм контролёр

д.п.н., к.т.н., профессор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И.Д. Рудинский

(подпись, дата)

Консультант по

экономическому разделу:

к.э.н., доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.В. Соловей

(подпись, дата)

Калининград

2018

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования «Калининградский государственный**

**технический университет»**

Факультет Автоматизации Производства и Управления

Кафедра Систем Управления и Вычислительной техники

Направление подготовки Прикладная информатика

**УТВЕРЖДАЮ**

Зав. кафедрой СУ и ВТ

к.т.н., доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Петрикин

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2018 г.

**ЗАДАНИЕ**

**на выпускную квалификационную работу студенту**

Саркисяну Эдуарду Валерьевичу

* Тема выпускной квалификационной работы: Автоматизированная система организации производственного процесса в студии создания графического контента. Утверждена приказом от 15 мая 2018 г. № 723ОЧ.
* Срок сдачи студентом выпускной квалификационной работы \_\_\_\_\_\_\_\_
* Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов):
* анализ объекта автоматизации и постановка задач ВКР;
* предлагаемые проектные решения;
* реализация и внедрение Системы;
* Узловой вопрос выпускной квалификационной работы: Разработка автоматизированной системы организации производственного процесса в студии создания графического контента.
* Перечень графического материала:
* организационная структура студии;
* функциональная структура студии;
* схемы технологических процессов обработки данных;
* примеры чертежей форм документов и видеокадров.
* Консультанты по выпускной квалификационной работе (с указанием относящихся к ним разделов):

|  |  |  |  |
| --- | --- | --- | --- |
| Раздел | Консультант | Подпись консультанта; дата | |
| Задание выдал | Задание принял |
| Экономическая эффективность создания Системы | к. э. н.  Соловей М. В. |  |  |

* Дата выдачи задания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_доцент / В.Г. Пожидаев /

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Э.В. Саркисян /

График выполнения защиты выпускной квалификационной работы по направлению 09.03.01 «Информатика и вычислительная техника»

|  |  |  |  |
| --- | --- | --- | --- |
| № | Дата | Вид работы | Согласовывает и отмечает выполнение |
| 1 | 10.05.2017 – 16.05.2018 | Преддипломная практика, уточнение темы выпускной квалификационной работы (ВКР) и перечня документов подлежащих разработке вопросов ВКР | Зав. кафедрой, к.т.н., доцент Петрикин В.А. |
| 2 | 17.05.2017 – 23.05.2018 | Выполнение ВКР | Зав. кафедрой, к.т.н., доцент Петрикин В.А. |
| 3 | 24.05.2017 – 30.05.2018 | Выполнение ВКР, получение задания по экономическому разделу | Зав. кафедрой, к.т.н., доцент Петрикин В.А., доцент Соловей М.В. |
| 4 | 31.05.2017 – 06.06.2018 | Выполнение ВКР, предоставление отчета по практике | Зав. кафедрой, к.т.н., доцент Петрикин В.А. |
| 5 | 07.06.2017 – 13.06.2018 | Выполнение ВКР | Зав. кафедрой, к.т.н., доцент Петрикин В.А. |
| 6 | 14.06.2018 – 17.06.2018 | Предоставление ВКР нормконтролеру, проверка на плагиат | Проф. Рудинский И.Д. |
| 7 | 18.06.2018 – 20.06.2018 | Оформление ВКР | Проф. Рудинский И.Д. |
| 8 | 21.06.2018 – 24.06.2018 | Предоставление ВКР на кафедру, оформление допуска к защите ВКР на кафедре и в деканате | Доц. Ломакина Г.В., завкафедрой Петрикин В.А., декан А.В. Калинин |
| 9 | 25.06.2018 – 05.07.2018 | Защита ВКР |  |

АННОТАЦИЯ

Данная пояснительная записка содержит описание результатов выполнения выпускной квалификационной работы.

Объектом выполнения работы выступает автоматизированная система организации производственного процесса в студии создания графического контента.

Процесс выполнения работы включает анализ объекта автоматизации, постановку задачи, разработку путей её решения, а также выбор средств и реализацию предложенных решений.

Результатом выполнения работы являются работоспособная автоматизированная система организации производственного процесса в студии создания графического контента на базе фреймворка «Qt» и комплект проектной документации на ее внедрение и эксплуатацию.

Пояснительная записка содержит 255 страниц, 12 иллюстраций, 30 таблиц, 7 приложений, 9 источников.

**ANNOTATION**

This explanatory note contains a description of the results of the final qualifying work.

The object of the work is an automated system for organizing the production process in the studio creating graphic content.

The process of performing the work includes the analysis of the object of automation, the formulation of the problem, the development of ways to solve it, as well as the choice of means and the implementation of the proposed solutions. The result of the work is a workable automated system for organizing the production process in the studio creating graphic content based on the "Qt" framework and a set of project documentation for its implementation and operation.

**Оглавление**

[1. АНАЛИЗ ОБЪЕКТА АВТОМАТИЗАЦИИ 13](#_Toc517225240)

[1.1. Общая характеристика студий 13](#_Toc517225241)

[1.2. Стратегические цели студий 14](#_Toc517225242)

[1.3. Бизнес-модель студии 15](#_Toc517225243)

[1.4. Организационная структура студий 16](#_Toc517225244)

[1.4.1. Отдел моделирования 18](#_Toc517225245)

[1.4.2. Отдел текстурирования 19](#_Toc517225246)

[1.4.3. Отдел VFX 19](#_Toc517225247)

[1.4.4. Отдел сценария и концепта 19](#_Toc517225248)

[1.4.5. Отдел сетапа 20](#_Toc517225249)

[1.4.6. Отдел анимации 20](#_Toc517225250)

[1.4.7. Отдел рендеринга 21](#_Toc517225251)

[1.4.8. Отдел композитинга 21](#_Toc517225252)

[2. ВЫЯВЛЕНИЕ ПРОБЛЕМ И ПОСТАНОВКА ЗАДАЧ 22](#_Toc517225253)

[2.1. Проблемы студии 22](#_Toc517225254)

[2.1.1. Проблемы организационного характера 22](#_Toc517225255)

[2.1.2. Проблемы рабочего характера 22](#_Toc517225256)

[2.1.3. Влияние проблем на деятельность студии 23](#_Toc517225257)

[2.2. Задачи для решения проблем 24](#_Toc517225258)

[3. ТРЕБОВАНИЯ К СИСТЕМЕ 25](#_Toc517225259)

[3.1. Требования к структуре и функционированию системы. 25](#_Toc517225260)

[3.2. Требования к персоналу 26](#_Toc517225261)

[3.3. Требования безопасности 27](#_Toc517225262)

[3.4. Требования к защите информации 28](#_Toc517225263)

[3.5. Требования по сохранности информации при авариях. 28](#_Toc517225264)

[3.6. Требования по стандартизации и унификации 28](#_Toc517225265)

[3.7. Требования к видам обеспечения 28](#_Toc517225266)

[3.7.1. Требования к информационному обеспечению 28](#_Toc517225267)

[3.7.2. Требования к лингвистическому обеспечению 29](#_Toc517225268)

[3.7.3. Требования к методическому обеспечению. 29](#_Toc517225269)

[3.8. Описание автоматизируемых функций 29](#_Toc517225270)

[3.8.1. Распределение функций 29](#_Toc517225271)

[4. ОБЗОР ГОТОВЫХ РЕШЕНИЙ 32](#_Toc517225272)

[4.1. Система управления производственным процессом «Alienbrain» 32](#_Toc517225273)

[4.1.1. Структура «Alienbrain» 32](#_Toc517225274)

[4.1.2. Возможности «Alienbrain» 33](#_Toc517225275)

[4.2. Системы голосовых конференций 34](#_Toc517225276)

[4.2.1. Приложение «Skype» 34](#_Toc517225277)

[4.2.2. Приложение «Teamspeak» 35](#_Toc517225278)

[4.2.3. Приложение «Discord» 35](#_Toc517225279)

[4.3. Выводы обзора готовых проектных решений 37](#_Toc517225280)

[4.3.1. Выводы обзора системы «Alienbrain» 37](#_Toc517225281)

[4.3.2. Выводы обзора систем голосовых конференций 37](#_Toc517225282)

[5. ПРЕДЛАГАЕМЫЕ ПРОЕКТНЫЕ РЕШЕНИЯ 38](#_Toc517225283)

[5.1. Сетевые характеристики взаимодействия приложений 38](#_Toc517225284)

[5.2. Поддерживаемые ОС 38](#_Toc517225285)

[5.3. Описание информационной базы системы 39](#_Toc517225286)

[5.4. Комплекс технических средств 40](#_Toc517225287)

[5.4.1. Структура комплекса технических средств 40](#_Toc517225288)

[5.4.2. Список технических средств 41](#_Toc517225289)

[5.4.3. Описание функционирования КТС 41](#_Toc517225290)

[5.4.4. Конфигурация комплекса технических средств 41](#_Toc517225291)

[5.4.5. Методы защиты технических средств 42](#_Toc517225292)

[5.4.6. Аппаратура передачи данных 42](#_Toc517225293)

[5.4.7. Организация обслуживания комплекса технических средств 42](#_Toc517225294)

[5.5. Необходимые инструменты для реализации 42](#_Toc517225295)

[6. ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ СОЗДАНИЯ «АСОППС» 44](#_Toc517225296)

[6.1. Этапы разработки системы 44](#_Toc517225297)

[6.2. Оценка затрат на разработку системы 45](#_Toc517225298)

[6.2.1. Оценка затрат на электроэнергию 46](#_Toc517225299)

[6.3. Преимущества реализации системы 47](#_Toc517225300)

[6.4. Экономическое преимущество внедрения системы в студию 47](#_Toc517225301)

[7. РЕАЛИЗАЦИЯ «АСОППС» 49](#_Toc517225302)

[7.1. Обоснование выбора объектно-ориентированного подхода и средств разработки 49](#_Toc517225303)

[7.1.1. Объектно-ориентированный подход 49](#_Toc517225304)

[7.1.2. Язык программирования «Python» 50](#_Toc517225305)

[7.1.3. Qt-фреймворк 52](#_Toc517225306)

[7.2. Классы системы 53](#_Toc517225307)

[7.2.1. Сущности системы 55](#_Toc517225308)

[7.2.2. Исполняющие механизмы 60](#_Toc517225309)

[7.2.3. Вспомогательные классы 61](#_Toc517225310)

[7.2.4. Графический интерфейс приложений 62](#_Toc517225311)

[8. Описание контрольного примера использования системы 64](#_Toc517225312)

[8.1. Подсистема «CGX Artist» 64](#_Toc517225313)

[8.1.1. Верхняя панель 65](#_Toc517225314)

[8.1.2. Панель представлений 65](#_Toc517225315)

[8.1.3. Форма представления проекта 66](#_Toc517225316)

[Приложение А. Бизнес-модель студий 72](#_Toc517225317)

[Приложение Б. Описание бизнес-процессов студии 84](#_Toc517225318)

[Приложение В. Описание автоматизируемых функций 151](#_Toc517225319)

[Приложение Г. Спецификации серверов 169](#_Toc517225320)

[Приложение Д. Листинг некоторых классов системы 172](#_Toc517225321)

[Приложение Е. Техническое задание на реализацию системы 214](#_Toc517225322)

Приложение Ж. Примеры графического интерфейса системы……………250

Введение

Эффективность деятельности предприятий в рыночных условиях напрямую зависит от оперативной работы всех его подразделений, скорости и качества выполнения различного рода работ, точности и своевременности расчетов с контрагентами.

Решение этих задач связано с необходимостью обработки значительных объемов информации в короткие сроки, организации деятельности многих людей, работающих с единым набором данных. Эффективным решением подобных задач является применение информационных систем, объединяющих в себе большую часть информационных процессов, происходящих на предприятии.

При внедрении систем автоматизации повышается уровень и скорость обслуживания, минимизируются ошибки и неточности при выполнении работ и расчетов с контрагентами, а менеджеры различных уровней и сам руководитель предприятия получают возможность в любой момент проанализировать как свою работу, работу своих подчиненных, так и работу всей компании.

Всё вышесказанное применимо и к сфере медийного бизнеса, который в настоящее время становится популярным в РФ. Даже на текущий момент, учитывая распространенность и многообразие информационных технологий, во многих студиях, занимающихся разработкой графического контента, сотрудники (художники) некорректным образом организовывают свою деятельность, ежедневно выполняют рутинную работу, которую следует поручить компьютеру, что значительно усложняет и тормозит общий процесс выполнения и внесение изменений в проект.

Такого рода предприятие действует как универсальная система, предоставляющая клиентам услуги в разработке видеороликов, изображений и прочих медиа-материалов.

Схемы функционирования предприятий данного характера очень схожи между собой и отличаются наличием тех или иных отделов или процессов производства. Данный факт позволяет не рассматривать конкретную студию и создать универсальную систему, которая будет легко внедряема на каждый объект.

В данной выпускной квалификационной работе (ВКР) описываются решения по созданию автоматизированной системы организации производственного процесса в студиях создания графического контента на базе фреймворка «Qt». Для простоты условимся, что студии, разрабатывающие графический контент, будем называть «CG-студии» или «студии», а разрабатываемую систему – «СG X Studio» или «АСОППС» (автоматизированная система организации производственного процесса в студии). Моделирование системы и бизнес-процессов разрабатывалось в пакете бизнес-моделирования «Business Studio». Всё отчеты, касающиеся модели, также были сгенерированы в данном пакете.

Данная ВКР содержит восемь глав:

1. «Анализ объекта автоматизации»;
2. «Выявление проблем и постановка задач»;
3. «Требования к системе»;
4. «Обзор готовых решений»;
5. «Предлагаемые проектные решения»;
6. «Технико-экономическое обоснование создания АСОППС»;
7. «Реализация АСОППС»;
8. «Описание контрольного примера использования системы».

Первая глава содержит результаты обследования предприятия, его организационную и функциональную структуру. Построена детальная функциональная модель основных рабочих процессов в нотации IDEF0, заданы параметры ее элементов.

Вторая глава содержит описание выявленных в ходе анализа предприятия проблем и постановку задач для их решения.

В третьей главе приведены требования к автоматизируемой системе.

В четвертой главе рассматриваются готовые проектные решения.

В пятой главе описываются предлагаемые проектные решения, а также необходимый инструментарий для реализации.

В шестой главе приведено экономическое обоснование реализации системы, а также экономическая эффективность внедрения разрабатываемого продукта;

В седьмой главе детально рассмотрена реализация автоматизированной системы организации производственного процесса в студиях.

В восьмой главе приведен пример использования системы.

# АНАЛИЗ ОБЪЕКТА АВТОМАТИЗАЦИИ

## Общая характеристика студий

CG-Студии – это предприятия, специализирующиеся на комплексном решении сложных задач в области компьютерной графики для кино и рекламы. Они сопровождают своих клиентов в течение всего цикла производства проекта: от разработки художественных решений для картины или рекламного ролика, создания аниматиков и превизуализации, супервайзинга и контроля съемки кадров под графику на площадке до постпродакшена и цветокоррекции.

Спектр решаемых студиями задач включает в себя разработку:

* концепт-арта;
* дизайна и превизуализации проекта;
* работу с визуальными эффектами (взрывы, разрушения, симуляции жидкостей, дымов, огня, тканей);
* анимацию объектов и персонажей;
* 2D и 3D композитинг;
* моушн-дизайн.

Для работы над проектами студии используют не только инструменты и программные пакеты, давно ставшие индустриальным стандартом, но и также могут развивать свои собственные разработки.

В РФ отсутствуют законы, регулирующие тематику и содержание конечного продукта студий компьютерной графики, поэтому руководители таких предприятий оперируют моральными принципами при выборе дизайна, стиля и темы проекта в качестве допустимого к производству.

Ниже приведен список нескольких крупных Российских студий.

Таблица 1.1 – Крупные Российские студии

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Местоположение | Направление | Персонал |
| ALGOUS Studio | Москва | Фильмы, реклама | До 100 |
| Amalgama VFX | Москва | Фильмы | Больше 100 |
| CGF | Москва | Фильмы, реклама, геймдев | До 100 |
| Asymmetric VFX | Санкт-Петербург | Мультфильмы, геймдев | Больше 100 |
| CineLab | Москва | Фильмы | До 100 |

## Стратегические цели студий

Стратегические цели представляют собой результаты, которых стремится достичь компания в перспективе.

В соответствии с методологией «системы сбалансированных показателей (Balanced Scorecard)» цели разбиваются на четыре блока:

* финансы;
* клиенты;
* бизнес-процессы;
* развитие и обучение.

Основной целью студии является увеличение чистой прибыли. Так как доход поступает только от проектной деятельности, каждая студия старается придерживаться конкурентоспособному качеству своего продукта, либо стать лидером в своей области.

Студий очень много, поэтому конкуренция заставляет поддерживать своих специалистов на высоком уровне, поэтому данный аспект является одной из важных целей.

Для обучения персонала требуется время и средства. Если сократить время выполнения проекта, можно выделить его на обучение начальников отделов или потенциальных художников, которые в дальнейшем позволят решать сложные задачи на порядок быстрее или смогут заниматься инновационными разработками, что в свою очередь тоже позволит увеличить эффективность выполнения проекта.

Все выше сказанное окажет влияние на улучшение качества конечного продукта студий, за счёт этого вырастет клиентская база и увеличится доход студии.

В приложении А приведена стратегическая карта студий.

## Бизнес-модель студий

Для описания информационных бизнес-процессов прекрасно подходит нотация IDEF0. Инструментом моделирования бизнес-процессов стала CASE-система «Business Studio». Все диаграммы и регламентирующие документы были сгенерированы в данном программном комплексе.

Диаграммы процессов приведены в приложении А, а регламенты в приложении Б.

Каждый документ описывает важные атрибуты процессов, такие как:

* содержание деятельности процесса;
* владельцев процесса;
* исполнителей процесса;
* начало выполнения процесса;
* результаты процесса;
* требования к срокам выполнения процесса;
* документация процессов;
* взаимодействие с другими процессами;
* описание подпроцессов.

Также в конце документов находится таблица состава набора объектов и ролей, участвующих в выполнении процесса.

Стоит отдельно описать главный процесс «А0 Создание графического контента», содержанием деятельности которого является разработка (по требованию заказчика) комплекса 3D-моделей, текстур, анимаций и симуляций, скомпонованных в единый видеоролик или же статическое изображение для применения в рекламе, кинофильме, мобильном приложении или видео игре. Владельцем процесса является директор – он несет полную ответственность перед заказчиком за конечный продукт. В роли исполнителей выступают менеджер и отдел разработки. Начало выполнения проекта определяется в общем техническом задании, согласованным с заказчиком.

## Организационная структура студий

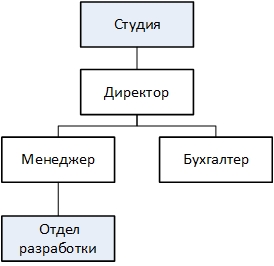


Рис. 1.1 – Организационная структура студий

Из рисунка 1.1 «Организационная структура студий» видно, что студии имеют не сложную структуру.

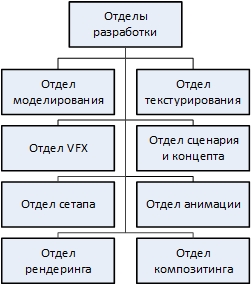


Рис. 1.2 – Неотъемлемая часть организационной структуры отдела разработки

Полная организационная структура отдела разработки приведена в приложении Б.

Стоит обратить внимание на отдел разработки, количество подотделов которого может варьироваться.

На рисунке 1.2 «Неотъемлемая часть организационной структуры» приведена структура подотделов, которая присутствует в отделе разработки каждой студии. Дополнительными подотделами (далее по тексту просто «отдел») могут стать отделы:

* инновационных разработок;
* моушн дизайна;
* специальных эффектов;
* звуковых эффектов;
* пластического грима.

Также студия может организовать не стандартный отдел для реализации задач, которые, по ее мнению, являются важнейшими при разработке проекта. Каждым отделом управляет начальник отдела.

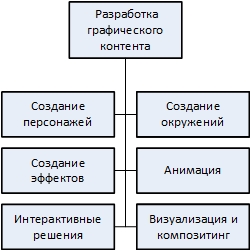


Рис. 1.3 – Функциональная структура студии

Функциональная структура студии, изображенная на рисунке 1.3, во многом совпадает с его организационной структурой.

### Отдел моделирования

Данный отдел по эскизам (концепт-артам) создает цифровые модели персонажей и элементов окружений. Именно так закладываются будущие возможности героев, их способности двигаться и переживать эмоции.

Специалисты отдела моделирования обладают объемным мышлением и в совершенстве владеют программным обеспечением. Цифровому скульптору необходимо не только добиться внешнего сходства модели с прообразом, но и технически правильно воплотить модель, учесть поведенческие особенности персонажа во время анимации, деформацию мускулов, тканей и много другое.

3D моделлеры работают на начальном этапе производственного цикла и создают фундамент для успешной и комфортной работы многих других отделов.

### Отдел текстурирования

Данный отдел в соответствии с эскизами (концепт-артами) рисуют текстуры для объектов окружений и персонажей. Технические требования, такие как размер текстур, формат, детализация, устанавливаются начальником отдела.

Для каждого объекта создается определенный набор текстур – текстуры цвета, рельефа, бликов, иногда используются фотографии поверхностей различных предметов. С их помощью специалисты имитируют физические свойства любого материала.

### Отдел VFX

VFX (от англ. *Visual Effects*) – визуальные эффекты. Данный отдел занимается разработкой визуальных эффектов, которые невозможно смоделировать или создать в реальной жизни. Например, невозможно создать разрушение целого города на киносъемочной площадке. Ниже приведены несколько примеры эффектов:

* взрывы;
* разрушения;
* симуляции жидкостей;
* симуляции дымов;
* симуляции огня;
* тканей;
* нетривиальное поведение частиц;
* световые эффекты.

### Отдел сценария и концепта

Данный отдел является ключевым на этапе производства. Он занимается разработкой сценария фильма, рекламы, игры или небольшого ролика и отрисовкой концептов в заданной заказчиком тематике и стилистике. Стоит отметить, что продукт отдела является регламентом (в плане визуального представления) для всех остальных отделов. Художники не могут создавать модели, текстуры, эффекты и многое другое на своё усмотрение, они должны строго придерживаться концепт-артам, созданным в данном отделе.

### Отдел сетапа

Данный отдел занимается подготовкой персонажей и предметов к анимации. Создают анимационный скелет, делают для него элементы управления, создают мускульную систему и привязывают ее к модели, а также подготавливают одежду к просчету коллизий и динамики. Для каждого персонажа необходим индивидуальный подход на сетапе, так как в зависимости от действий, которые по сюжету совершает персонаж в кадре, используются различные комплексы технологических решений.

Одна из основных задач сетапа-художника – это достижение удобства работы аниматора с моделью. Отделы анимации и сетапа работают в тесном сотрудничестве.

### Отдел анимации

Данный отдел занимается анимированием персонажей и объектов. Используются все существующие методы создания анимации:

* key frames (ручной метод);
* motion capture (захват движения).

На первом этапе анимация еще сырая и требует доработки.

На втором этапе прорабатывается поведение одежды персонажей, волос, меха, тканей, окружающих объектов, взаимодействующих друг с другом в соответствии с законами физики.

На третьем этапе (завершающим) производится доскональная проверка анимации в каждом кадре – cleanup. Проверяется каждая сцена на предмет наличия ошибок в анимации, нестыковок по монтажу, наличия всех объектов в сцене и многих других мелочей.

### Отдел рендеринга

Данный отдел занимается постановкой света в сцене, настройкой материалов и шейдеров, а также производит конечную визуализацию сцены. Отдел текстурирования функционально подчиняется отделу рендеринга, так как все текстуры используются в материалах объектов или в создаваемых шейдерах. Главной целью отдела является сырая, но максимально качественная картинка, которая отправится в отдел композитинга на ручную доработку для придания её большей реалистичности.

### Отдел композитинга

Данный отдел собирает все отрендеренные сцены, дорабатывает их, создаёт 2D визуальные эффекты, компонует всё с отснятым материалом и производит финальную цветокоррекцию. После цветокоррекции монтажёр готовит финальный видеоролик, и вставляет в него звуковую дорожку.

# ВЫЯВЛЕНИЕ ПРОБЛЕМ И ПОСТАНОВКА ЗАДАЧ

## Проблемы студии

Проанализировав модель студии и диаграммы рабочих процессов сотрудников (приложение А), были выявлены 2 группы проблем:

* организационные;
* рабочие.

### Проблемы организационного характера

Данная группа проблема связана с созданием и настройкой проекта, а также с передачей данных между сотрудниками отдела разработки.

К этой группе относятся следующие проблемы:

* трудоемкое создание и изменение структуры проекта на серверах отделов;
* трудоемкое распределение задач по художникам и контроль проделанной работы (начальниками над работой художников);
* трудоемкая передача данных между сотрудниками студии;
* отсутствии возможности быстрого доступа к готовым асетам и результатам проекта;
* отсутствие моментальной связи между художниками и начальниками.

### Проблемы рабочего характера

Данная группа проблема связана с рабочей деятельностью художников и начальников всех отделов.

К этой группе относятся следующие проблемы:

* длительное создание файловой структуры локального проекта;
* частые сбои и вылет используемых приложений;
* долгая настройка приложения перед его запуском;
* отсутствие логической связи между настройками используемых приложений и текущим проектом;
* отсутствие автоматизированного инструментария управления личными ресурсами и ресурсами локального проекта.

### Влияние проблем на деятельность студии

Рассмотрим более детально основные выявленные проблемы, приводящие к различного рода потерям.

Во-первых, из-за трудоемкого создания и изменения структуры проекта на серверах отделов невозможно быстро вносить изменения в проект, что приводит к длительному перераспределению файлов проекта. Это приводит к остановке производства более чем 20 минут.

Во-вторых, каждый художник использует большое количество различных приложений в своей рабочей деятельности. Каждое из приложений имеет массу параметров, которые художник настраивает под себя, из этого следует, что после запуска приложения (иногда и перед запуском) нужно уделить около 5 минут на этот процесс. Отрицательной особенность является то, что все эти приложений часто аварийно завершают работу и после повторного запуска требуется настройка параметров. Сохранять настроенные параметры нельзя, так как на рабочем месте могут работать до трех художников (в разные смены). Так же художник вынужден тратить около 5 минут на создание файловой структуры своего локального проекта, описательные создавать файлы для задач.

В-третьих, художники часто консультируются друг у друга и у начальника по вопросам различного рода, для это им приходится покидать свое рабочее место, либо использовать социальные сети, но этот подход не эффективен, потому что социальные сети не оповещают в должной форме о новых сообщениях и не позволяют продемонстрировать свой экран. Из-за этого часто прерывается рабочий процесс многих сотрудников.

В-четвертых, менеджер не имеет доступа к готовым асетам проекта, для презентации заказчику. Менеджер должен отвлекать начальников отдела разработки, чтобы получить финальный результат проекта, или же визуализацию асета.

В совокупности из-за всех выявленных проблем сотрудники студии часто выполняют рутинную работу, которая сильно тормозит разработку проекта.

## Задачи для решения проблем

Для решения выявленных проблем необходимо реализовать систему, которая обеспечила бы:

* автоматизированное создание и настройку проекта на серверах отдела;
* общий доступ к исходным данным проекта для каждого начальника;
* хранение данных проекта в виде ассетов;
* управление личными ресурсами художникам;
* возможность внутри студийных голосовых конференций;
* кастомизацию рабочих мест художников;
* автоматизирование рутинных процессов;
* быстрый просмотр готовых асетов и результатов проекта.

# ТРЕБОВАНИЯ К СИСТЕМЕ

## Требования к структуре и функционированию системы.

Система должна быть децентрализованной, то есть состоять из:

* трех подсистем;
* одного модуля;
* одного самостоятельного приложения.

Таблица 3.1 – Компоненты системы

|  |  |
| --- | --- |
| Модуль | Пояснение |
| CGX Pipeline | Конвейер передачи данных |
| Подсистема | Пояснение |
| CGX Artist | АРМ художника |
| CGX Supervisor | АРМ начальника отдела |
| CGX Manager | Мобильное приложение менеджера |
| Приложение | Пояснение |
| CGX Conference | Приложение для голосовых конференций |

Все подсистемы и модуль являются несамостоятельными приложениями, которые должны запускаться на разных машинах. На рисунке 3.1 приведены зависимости всех компонентов системы.

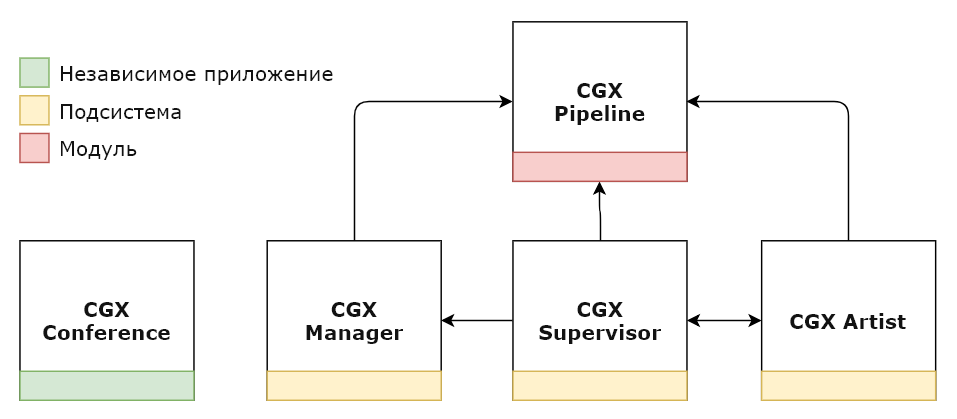


Рис. 3.1 – Диаграмма зависимостей компонентов системы

Таблица 3.2 – Предназначения компонентов системы

|  |  |
| --- | --- |
| Компонент системы | Реализует |
| CGX Pipeline | связь между подсистемами |
| общий доступ к данным |
| CGX Artist | управление личной библиотекой |
| настройку программ под проект |
| управление локальным инстансом проекта |
| взаимодействие с задачами проекта |
| CGX Supervisor | управление локальным проектом |
| управление задачами проекта |
| контроль выполнения проекта |
| CGX Manager | доступ к готовым асетам и результатам проекта |
| CGX Conference | внутристудийное общение |

## Требования к персоналу

Пользователи требуемой системы должны быть разделены на 3 группы:

* начальники отделов;
* художники;
* менеджеры.

Для всех групп пользователей требуются базовые навыки работы на ПК и чёткое понимание студийного производственного процесса.

## Требования безопасности

Защита технических средств от воздействий электрического тока, электромагнитных полей, акустических шумов и т.п. должна осуществляться в соответствии с требованиями по эксплуатации, предъявляемыми к оборудованию его разработчиками.

Устанавливаемое оборудование должно соответствовать требованиям электробезопасности по ГОСТ Р МЭК 60065-2002.

Допустимый уровень электромагнитных полей на рабочих местах должен отвечать требованиям ГОСТ 12.1.006-84.

Факторы, оказывающие вредные воздействия на здоровье, (в том числе инфракрасное, ультрафиолетовое, рентгеновское и электромагнитное излучения, вибрация, шум, электростатические поля, ультразвук строчной частоты и т.д.), не должны превышать действующих норм (СанПиН 2.2.2. /2.4.1340-03 от 03.06.2003 г. с изменениями от 25 апреля 2007 г.).

Комплекс технических средств должен соответствовать требованиям техники безопасности, основными из которых являются:

* все внешние элементы технических устройств, находящиеся под напряжением, должны иметь защитное заземление;
* технические устройства должны быть установлены в местах, обеспечивающих свободный и безопасный доступ к ним при эксплуатации и проведении профилактического обслуживания;
* сотрудники, которые работают на технических средствах, должны проходить обучение, инструктаж, проверку знаний правил, норм и инструкций по технике безопасности;
* в помещении, предназначенном для эксплуатации технических средств, должны быть обеспечены противопожарные меры безопасности.

## Требования к защите информации

Работа с системой должна осуществляться только после аутентификации. Регистрация художника и начальника происходит на локальной машине (на той, где установлено приложение).

## Требования по сохранности информации при авариях.

В приложении управления проектом должен быть предусмотрен механизм резервного копирования текущих проектов. Процедура резервного копирования должна производиться начальниками отделов вручную 1 раз в конце рабочего дня для незавершенных проектов.

## Требования по стандартизации и унификации

Система должна быть кроссплатформенной, а стилистика приложений не должна зависеть от платформы. Ниже приведена таблица соответствия приложения и операционных систем, на которых оно должно работать.

## Требования к видам обеспечения

### Требования к информационному обеспечению

Все данные системы должны храниться на физическом носителе «Жёсткий диск», подключенном к серверному компьютеру, в структурированном виде под управлением реляционной СУБД. Исключения составляют файлы данных, предназначенные для просмотра (изображения, анимации, документы и т.п.). Такие файлы сохраняются в файловой системе, а в БД размещаются ссылки на них.

### Требования к лингвистическому обеспечению

Графические интерфейсы приложений должны быть выполнены на русском и английском языках.

### Требования к методическому обеспечению.

В рамках выбора системы должны быть учтены соответствующие административные регламенты, в которых определены процессы деятельности и функции сотрудников, их права, обязанности и ответственность при использовании системы.

Приложения должны включать справочную систему и подсказки в графическом интерфейсе.

## Описание автоматизируемых функций

Согласно этой структуре, Система должна состоять из 5 приложений, каждое из которых содержит комплекс функций.

Схема функциональной структуры и полное описание автоматизируемых функций приведены в приложении В.

### Распределение функций

**Подсистема «CGX Supervisor»**. Данная подсистема предназначена начальникам отделов. Она взаимодействует с подсистемой «CGX Artist» через модуль «CGX Pipeline» и реализует следующие функции:

* быстрая фильтрация, сортировка и поиск по проекту;
* отсылка готового результата на главный сервер;
* отсылка файлов своим подчиненным;
* подключение к существующему проекту;
* просмотр активных художников;
* управление задачами своего отдела.

**Подсистема «CGX Artist»**. Данная подсистема предназначена художникам отделов. Она взаимодействует с подсистемой «CGX Supervisor» через модуль «CGX Pipeline» и реализует следующие функции:

* создание личного аккаунта;
* автоматическая настройка переменных проекта;
* быстрая фильтрация, сортировка и поиск по личным ресурсам и ресурсам проекта;
* взаимодействие с задачами проекта;
* запуск приложений в окружении проекта;
* быстрый просмотр hdr и exr файлов;
* определение пользовательских настроек;
* отсылка файлов коллегам по отделу;
* подключение к существующему проекту;
* подключение личной библиотеки к аккаунту.

**Подсистема «CGX Manager»**. Данная подсистема предназначается менеджеру отдела разработки, она дает ему возможность создать проект, над которым будет работать его отдел, просматривать асеты проекта и финальные результаты. Так же эта подсистема позволит ему отправлять скриншоты асетов и готовые результаты заказчику.

Функции подсистемы:

* отправка данных заказчику;
* просмотр асетов;
* просмотр готовых результатов.

**Приложение «CGX Conference»**. Мессенджер с поддержкой видеоконференций для пользователей ПК и гаджетов. Данное приложение способно организовывать голосовые конференции с настройкой канала связи и работать по принципу push-to-talk (нажми, чтобы сказать), создавать публичные и приватные чаты для обмена текстовыми сообщениями.

Функции приложения:

* Организация голосовых конференций
* Приватные чаты

**Модуль «CGX Pipeline»**. Данный модуль обеспечивает связь между подсистемами.

# ОБЗОР ГОТОВЫХ РЕШЕНИЙ

Перед обзором готовых решений разделим рассматриваемые системы на два типа:

* системы организации производственного процесса в студии;
* системы организации голосовых конференций.

На сегодняшний день на рынке существует множество программных пакетов, позволяющих организовывать голосовые конференции, ниже будут рассмотрены некоторые из них.

С системами организации производственного процесса дела обстоят хуже – имеется лишь один вариант программного обеспечения – «Alienbrain» от компании «Avid».

## Система управления производственным процессом «Alienbrain»

Данная система разработана компанией «Avid» и позволяет сотрудникам студии осуществлять полный контроль над выполнением проекта.

Цена на студию из 10 сотрудников: 99 000 $.

### Структура «Alienbrain»

Данная система состоит из двух подсистем:

* «Alienbrain server»;
* «Alienbrain client».

Серверное приложение служит хранилищем данных проекта, а также связывает клиентов между собой.

Клиентское приложение является обозревателем контента, находящегося на сервере.

### Возможности «Alienbrain»

#### Централизованное хранение файлов

Все файлы проекта располагаются на центральном сервере студии, где художники могут просматривать их, извлекать или блокировать для редактирования. Данные функции осуществляются через клиентские приложения.

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Все художники знают, где находятся важные файлы, и кто работает над ними в текущий момент. |  |
| Файлы никогда не удаляются случайно и не перезаписываются |  |

#### Контроль версий асетов

Сохраняется история версий всех асетов проекта. Возможно отслеживать изменения и, при необходимости, «откатываться» в любое время.

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Уверенность в правильной версии асета |  |

#### Устойчивость к тяжелым нагрузкам

Большинство цифровых систем управления производственным процессом, рассчитаны на работу с небольшим количеством легких файлов. «Alienbrain» был разработан для совместной работы больших коллективов над массивными проектами, поэтому он продолжает стабильно функционировать даже при высоких нагрузках.

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Стабильная передача данных между сотрудниками |  |

#### Управление проектами и его конфигурациями

«Alienbrain» дает администраторам полный контроль над проектами: от установки разрешений и привилегий, до модификации большого количества файлов, для резервного копирования и архивирования старого контента.

Можно управлять одним или несколькими проектами одновременно. Расширенные инструменты управления конфигурацией проекта позволяют легко поддерживать несколько преднастроенных проектов одновременно без дублирования его содержимого.

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Гибкая организация конфигураций проектов |  |

#### Настройка, автоматизация, интеграция.

Наличие набора интерфейсов прикладного программирования (API) позволяют программно получать доступ к контенту и метаданным.

|  |  |
| --- | --- |
| Плюсы | Минусы |
| Возможность эффективно автоматизировать общие задачи и настраивать инструменты, представления и поведение «Alienbrain» в соответствии с конкретными потребностями проекта. | Высокое требование к навыкам программирования |
|  | Долгое изучение API «Alienbrain» |

## Системы голосовых конференций

### Приложение «Skype»

Бесплатное программное обеспечение с закрытым кодом, обеспечивающее текстовую, голосовую и видеосвязь через Интернет между компьютерами (IP-телефония), опционально используя технологии пиринговых сетей, а также платные услуги для звонков на мобильные и стационарные телефоны.

Программа также позволяет совершать конференц-звонки (до 25 голосовых абонентов, включая инициатора), видеозвонки (в том числе видеоконференции до 10 абонентов), а также обеспечивает передачу текстовых сообщений (чат) и передачу файлов. Есть возможность вместе с изображением с веб-камеры передавать изображение с экрана монитора, а также создавать и отправлять видеосообщения пользователям настольных версий программы.

### Приложение «Teamspeak»

Программа, предназначенная для голосового общения в сети Интернет посредством технологии VoIP. От классического телефона отличается практически неограниченным количеством абонентов, разговаривающих одновременно. Больше всего это похоже на многоканальную рацию, в которой можно одновременно пользоваться несколькими каналами. При этом доступны все опции, разработанные ранее для удобства использования раций в полевых (боевых) условиях. Программы этого типа предназначены прежде всего для геймеров, но могут использоваться везде, где необходима голосовая связь и координация большой группы людей.

Стоит отметить, что данное приложение работает только с аудио-данными и не имеет возможность демонстрации экрана.

### Приложение «Discord»

Бесплатный мессенджер с поддержкой VoIP и видеоконференций, изначально ориентированный для пользователей компьютерных игр. Настольное клиентское приложение реализовано для Windows (работает на Windows 7 и новее), macOS и Linux, мобильное приложение – для Android и iOS, существует также веб-клиент. Поддерживает 27 языков, включая русский.

Единая кодовая база для настольных, веб- и мобильных клиентов обеспечивается благодаря использованию фреймворка Electron. Серверы мессенджера размещены в 11 центрах обработки данных в разных частях мира. Также возможна организация локального сервера.

Приложение способно:

* организовывать голосовые конференции с настройкой канала связи и работать по принципу push-to-talk;
* создавать публичные и приватные чаты для обмена текстовыми сообщениями, а также оно обладает защитой от DDoS-атак.

Программа имеет браузерную версию, отличием от настольной версии является то, что режим push-to-talk работает только если вкладка с программой в фокусе.

При включении режима «стример» скрывается вся личная информация, отключаются звуки и уведомления на рабочий стол.

Включается режим автоматически (при запуске программ для трансляций, например, OBS Studio[en]), также режим можно включить или отключить вручную.

Для аудио используется кодек Opus, который имеет возможности эхоподавления, подавления посторонних шумов и автоматической регулировкой усиления. Для видео используется кодек VP8.

Поддерживается назначение «горячих клавиш».

Реализована функция отключения уведомлений, возможно отключить уведомления на мобильное устройство при бездействии на компьютере. Есть возможность включить озвучивание уведомлений.

В текстовом чате поддерживается присоединение файлов, картинок, вставка ссылок (для некоторых сайтов работает предпросмотр), форматирование текста и смайлики, размер вложения ограничен объёмом 8 МБ.

## Выводы обзора готовых проектных решений

### Выводы обзора системы «Alienbrain»

Данная система подходит для реализации практически всех поставленных задач. Очень гибкое проектное решение, так как разработчики данного пакета предоставляют API, которое можно использовать, как и в разрабатываемых приложениях, так и в скриптах.

Минусами данной системы являются:

* отсутствие возможности организации голосовых конференций;
* отсутствие запуска программ с определенными переменными окружения.

Но, так как выше было сказано про открытое API, данные проблемы решаемы.

### Выводы обзора систем голосовых конференций

Из рассмотренных выше приложений «Discord» является отличной готовой системой для решения поставленных коммуникационных задач. Данный программный комплекс будет интегрирован в систему «CGX Studio».

# ПРЕДЛАГАЕМЫЕ ПРОЕКТНЫЕ РЕШЕНИЯ

Техническое задание на реализацию системы находится в приложении Е.

## Сетевые характеристики взаимодействия приложений

Так как система должна быть децентрализованной, в качестве протокола взаимодействия между приложениями на транспортно-сетевом уровне необходимо использовать протокол TCP/IP. Для организации информационного обмена между приложениями также должен использоваться протокол TCP/IP.

В таблице 5.1 приведены соответствия приложений с понятиями «клиент-сервер»:

Таблица 5.1 – Клиент-серверные компоненты системы

|  |  |
| --- | --- |
| Клиент | Сервер |
| CGX Artist | CGX Pipeline |
| CGX Supervisor |  |
| CGX Manager |  |
| CGX Conference |  |

## Поддерживаемые ОС

Так как система должна быть кроссплатформенной, ниже приведена таблица соответствия приложений и операционных систем, на которых они должны работать.

Таблица 5.2 – Поддерживаемые операционные системы

|  |  |
| --- | --- |
| Приложение | ОС |
| CGX Pipeline | Linux Mint |
| KUbuntu. |
| CGX Artist | Windows 10 |
| Linux Mint |
| Ubuntu |
| CGX Supervisor | Windows 10 |
| Linux Mint |
| Ubuntu |
| Android (от 8.0 и выше) |
| iOS (от 9.3.5 и выше) |
| CGX Manager | Android (от 8.0 и выше) |
| iOS (от 9.3.5 и выше) |
| CGX Conference | Windows 10 |
| Linux Mint |
| Ubuntu |
| Android (от 8.0 и выше) |
| iOS (от 9.3.5 и выше) |

## Описание информационной базы системы

Все данные системы будут храниться на физическом носителе «Жёсткий диск», подключенном к серверному компьютеру, в структурированном виде под управлением реляционной СУБД SQLite 3.0. Исключения составляют файлы данных, предназначенные для просмотра (изображения, анимации, документы и т.п.). Такие файлы сохраняются в файловой системе, а в БД размещаются ссылки на них.

## Комплекс технических средств

### Структура комплекса технических средств

Техническую составляющую будут выполнять локальные сервера отделов студии и главный сервер.

Сервера отделов отвечают за передачу данных между начальником отдела и его подчиненными, а также отправку на главный сервер. К главному серверу подключаются сервера отделов и Wi-Fi роутер, который обеспечивает беспроводное соединение приложения «CGX Manager».

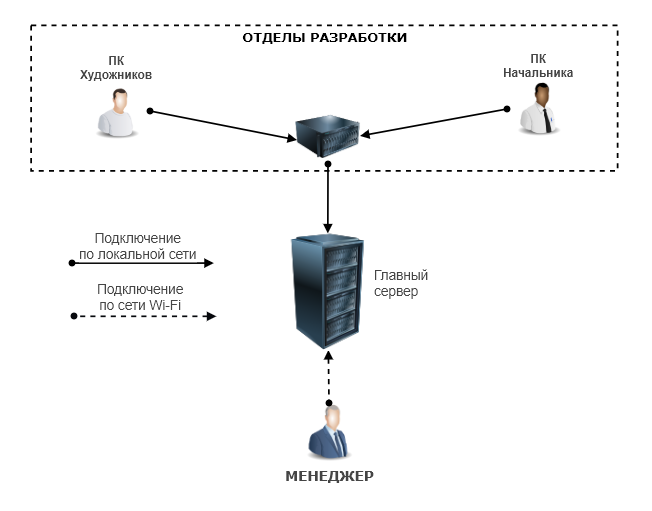


Рис. 5.1 – Схема структуры комплекса технических средств

### Список технических средств

* сервера отделов разработки;
* главный сервер;
* ПК начальников отдела;
* ПК художников;
* планшет менеджера.

### Описание функционирования КТС

При нормальном режиме работы все сервера работают непрерывно. Плановая перезагрузка делается автоматически каждую неделю, в 4:00, каждое воскресенье.

При пусковом режиме работы определяется целостность всех модулей серверов и разделов жестких дисков. Все приложения системы доступны для работы только после завершения пускового режима.

При перезагрузке производится быстрая проверка сервера. Проверка сервера осуществляется утилитой «Active Directory Health Profiler».

При аварийном режиме работы, например, при отключении электричества, происходит плавное отключение серверов благодаря наличию устройства бесперебойного питания. После устранения проблем происходит полная проверка целостности всех модулей серверов и разделов жесткого диска.

Описание решений по размещению КТС на объекте внедрения

Так как студия имеет серверную комнату готовую к эксплуатации, все сервера будут размещаться в ней.

### Конфигурация комплекса технических средств

Спецификации требуемого комплекса технических средств приведены в приложении Г.

### Методы защиты технических средств

Для защиты серверов от проблем с электричеством предусмотрено использование источника бесперебойного питания.

### Аппаратура передачи данных

Для успешного функционирования системы требуется обеспечение пропускной способности сети не менее 10 Гбит/c, чтобы не возникало длительных (свыше 5 мин.) задержек передачи данных больших размеров (до 200 ГБайт).

### Организация обслуживания комплекса технических средств

Для своевременного и качественного облуживания системы предусмотрено наличие администратора. Администратор должен обладать квалификацией инженера АСОИУ, а также навыками настройки и администрирования серверов, знаниями таких технологий, как ЯП Python и SQL.

Учитывая, что создателем данной системы является студент, и в дальнейшем производить её сопровождение не сможет, поддержкой будут заниматься работники, нанятые студией.

## Необходимые инструменты для реализации

Языком реализации приложений должен стать «С++» или «Python» с использованием фреймворка «Qt». Графический интерфейс будет написан на языке QML.

Для разработки системы потребуется среда разработки QtCreator в случае выбора языка «С++». Если же выбор падёт на язык Python, то необходимыми станут:

* среда разработки PyCharm от компании Jet Brains
* интерпретатор языка Python 3.6;
* библиотека PyQt 5.10;
* библиотека xnetwork;
* библиотека send2trash;
* библиотека PyInstaller 3.0.

# ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ СОЗДАНИЯ «АСОППС»

## Этапы разработки системы

В соответствии с ГОСТ 34.601-90 «Автоматизированные системы. Стадии создания» сформирован план работ по созданию «АСОППС», который представлен в виде таблицы.

Таблица 6.1 – Этапы разработки системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Стадии | Исполнитель | Начало | Окончание | Длительность |
| Формирование требований к системе | Системный аналитик, Системный архитектор | Пт. 01.06.18 | Чт. 05.06.18 | 4 дней |
| Разработка концепции системы | Системный аналитик | Пт. 08.06.18 | Пн. 11.06.18 | 3 дней |
| Формирование технического задания | Системный архитектор | Вт. 12.06.18 | Пн. 17.06.18 | 5 дней |
| Эскизное проектирование | Разработчик | Вт 19.06.18 | Чт. 24.06.18 | 5 дней |
| Разработка технического проекта | Программист | Пт. 22.06.18 | Чт. 12.07.18 | 20 дней |
| Ввод в действие | Тестировщик | Пт. 29.06.18 | Вт 01.07.18 | 2 дней |

Планирование проекта производилась в среде управления проектами Microsoft Project 2013. Диаграмма Ганта, наглядно отражающая сроки и последовательность реализации каждого из этапов работ представлена на рис. 6.1

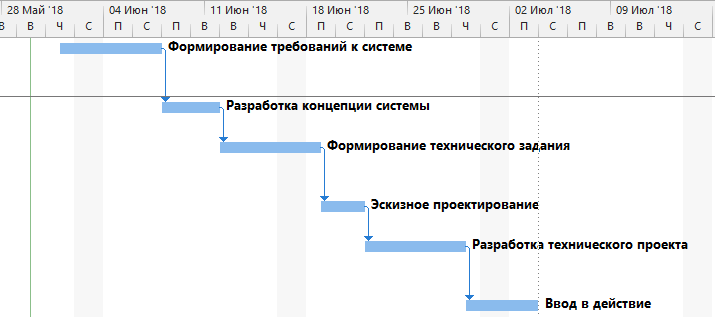


Рис. 6.1 – Диаграмма Ганта стадий создания «АСОППС»

## Оценка затрат на разработку системы

В таблице 6.2 приведены стандартные ставки персонала, необходимого для разработки, тестирования, внедрения и сопровождения системы.

Таблица 6.2 – Стандартные ставки разработчиков системы

|  |  |
| --- | --- |
| Должность | Стандартная ставка |
| Системный аналитик | 1 000,00 ₽/ч |
| Системный архитектор | 1 000,00 ₽/ч |
| Разработчик | 700,00 ₽/ч |
| Программист | 900,00 ₽/ч |
| Тестировщик | 800,00 ₽/ч |

Единовременные затраты на разработку и внедрение системы представлены в таблице 6.3

Таблица 6.3 – Затраты на разработку и внедрение системы

|  |  |  |  |
| --- | --- | --- | --- |
| Этап разработки | Длительность | Исполнительные лица | Затраты |
| Формирование требований к системе | 4 дней | Системный аналитик; Системный архитектор | 64 000,00 ₽ |
| Разработка концепции системы | 3 дней | Системный аналитик | 24 000,00 ₽ |
| Формирование технического задания | 5 дней | Системный архитектор | 40 000,00 ₽ |
| Эскизное проектирование | 5 дней | Разработчик | 28 000,00 ₽ |
| Разработка технического проекта | 20 дней | Программист | 144 000,00 ₽ |
| Ввод в действие | 2 дней | Тестировщик; Разработчик | 24 000,00 ₽ |
| **ИТОГО** | **39 дней** | **324 000,00 ₽** | |

### Оценка затрат на электроэнергию

Затраты на электроэнергию вычисляются по формуле:

SЭЛ.ЭН = РПОТ \* Т \* C, где

|  |  |
| --- | --- |
| SЭЛ.ЭН | затраты на электроэнергию |
| РПОТ | потребляемая мощность – 0.4 киловатт в час |
| Т | фонд времени за период амортизации – кол-во дней \* 8 часов |
| С | стоимость 1 киловатта энергии – 3,92 рублей |

Составим таблицу расчета затрата электроэнергии для каждого этапа разработки системы.

Таблица 6.4 – Затраты электроэнергии на разработку

|  |  |  |  |
| --- | --- | --- | --- |
| Этап разработки | Длительность | Формула расчета | Результат |
| Формирование требований к системе | 4 дней | SЭЛ.ЭН = 2 **∙** 0,4 **∙** 32 **∙** 3,92 | 101,00 ₽ |
| Разработка концепции системы | 3 дней | SЭЛ.ЭН = 0,4 **∙** 24 **∙** 3,92 | 38,00 ₽ |
| Формирование технического задания | 5 дней | SЭЛ.ЭН = 0,4 **∙** 40 **∙** 3,92 | 63,00 ₽ |
| Эскизное проектирование | 5 дней | SЭЛ.ЭН = 0,4 **∙** 40 **∙** 3,92 | 63,00 ₽ |
| Разработка технического проекта | 20 дней | SЭЛ.ЭН = 0,4 **∙** 160 **∙** 3,92 | 251,00 ₽ |
| **ИТОГО** | **39 дней** | **526,00 ₽** | |

Таким образом, общая сумма затрат на реализацию системы составит **324 526,00** рублей.

## Преимущества реализации системы

В главе «Обзор готовых решений» была рассмотрена система «Alienbrain», которая реализует практически все требуемые задачи. Цена данного продукта на годовую лицензию для студии с составом 50 сотрудников составит 495 000 $ (31 007 790 рублей на момент 30.05.2018). С учетом того, что размеры студии могут превышать число 50, цена на «Alienbrain» будет выше.

Данный аспект позволяет рассматривать реализацию собственной системы более выгодным решением, поскольку стоимость реализации на два порядка ниже и не зависит от количества сотрудников студии.

## Экономическое преимущество внедрения системы в студию

Разрабатываемая система позволит снизить время разработки проекта, что станет ключевым фактором увеличения дохода.

Появившееся время можно использовать для:

* повышения квалификации персонала;
* увеличения количества выполняемых проектов.

Повышение квалификации персонала является более эффективным шагом, данный аспект позволит сотрудникам решать более сложные задачи, делать более эффектные визуализации, но также данное решение потребует инвестиций и время.

# РЕАЛИЗАЦИЯ «АСОППС»

## Обоснование выбора объектно-ориентированного подхода и средств разработки

### Объектно-ориентированный подход

Объектно-ориентированное программирование (ООП) – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

Идеологически ООП – подход к программированию как к моделированию информационных объектов, решающий на новом уровне основную задачу структурного программирования: структурирование информации с точки зрения управляемости, что существенно улучшает управляемость самим процессом моделирования, что в свою очередь особенно важно при реализации крупных проектов.

Управляемость для иерархических систем предполагает минимизацию избыточности данных (аналогичную нормализации) и их целостность, поэтому созданное удобно управляемым – будет и удобно пониматься. Таким образом через тактическую задачу управляемости решается стратегическая задача – транслировать понимание задачи программистом в наиболее удобную для дальнейшего использования форму.

Основные принципы структурирования в случае ООП связаны с различными аспектами базового понимания предметной задачи, которое требуется для оптимального управления соответствующей моделью:

* **абстрагирование** для выделения в моделируемом предмете важного для решения конкретной задачи по предмету, в конечном счете – контекстное понимание предмета, формализуемое в виде класса;
* **инкапсуляция** для быстрой и безопасной организации собственно иерархической управляемости: чтобы было достаточно простой команды «что делать», без одновременного уточнения как именно делать, так как это уже другой уровень управления;
* **наследование** для быстрой и безопасной организации родственных понятий: чтобы было достаточно на каждом иерархическом шаге учитывать только изменения, не дублируя все остальное, учтенное на предыдущих шагах;
* **полиморфизм** для определения точки, в которой единое управление лучше распараллелить или наоборот – собрать воедино.

То есть фактически речь идет о прогрессирующей организации информации согласно первичным семантическим критериям: «важное/неважное», «ключевое/подробности», «родительское/дочернее», «единое/множественное». Прогрессирование, в частности, на последнем этапе дает возможность перехода на следующий уровень детализации, что замыкает общий процесс.

Обычный человеческий язык в целом отражает идеологию ООП, начиная с инкапсуляции представления о предмете в виде его имени и заканчивая полиморфизмом использования слова в переносном смысле, что в итоге развивает выражение представления через имя предмета до полноценного понятия-класса.

### Язык программирования «Python»

Python это язык программирования общего назначения, нацеленный в первую очередь на повышение продуктивности самого программиста, нежели кода, который он пишет. Говоря простым человеческим языком, на Python можно написать практически что угодно (веб-/настольные приложения, игры, скрипты по автоматизации, комплексные системы расчёта, системы управления жизнеобеспечением и многое многое другое) без ощутимых проблем. Более того, порог вхождения низкий, а код во многом лаконичный и понятный даже тому, кто никогда на нём не писал. За счёт простоты кода, дальнейшее сопровождение программ, написанных на Python, становится легче и приятнее по сравнению с Java или C++. А с точки зрения бизнеса это влечёт за собой сокращение расходов и увеличение производительности труда сотрудников.

Одним из составляющих успеха любой технологии является сообщество, созданное вокруг неё. Именно оно предопределяет будущий вектор развития путём совместных усилий.

Сообщество вокруг Python одно из самых сильных в мире IT. Это сложный хорошо организованный и постоянно развивающийся организм. Помимо сотни тысяч индивидуальных разработчиков и небольших компаний, Python поддерживают такие гиганты IT как:

* Google;
* Dropbox;
* Mozilla;
* Facebook;
* Yandex;
* Red Hat;
* Microsoft (с недавних пор очень активно, в частности с Visual Studio);
* Intel (активно ведёт исследовательскую работу в области параллельных вычислений на Python).

Данный факт свидетельствует о том, что крупные корпорации не боятся строить свой бизнес вокруг Python, они уверены в том, что технология будет жить, а, следовательно, проблем с поиском специалистов ждать не стоит. Более того, разнообразие приложений также радует, что свидетельствует о широком круге задач, которые Python решает мастерски.

### Qt-фреймворк

Многие привыкли считать, что Qt – лишь средство для создания только интерфейса пользователя. Это не так – Qt представляет собой полный инструментарий для программирования, который состоит из отдельных модулей и предоставляет:

* поддержку двух- и трехмерной графики (фактически, являясь стандартом для платформонезависимого программирования на OpenGL);
* возможность интернационализации, которая позволяет значительно расширить рынок сбыта программ;
* использование формата XML (extensible Markup Language);
* поддержку стандартных протоколов ввода/вывода;
* классы для работы с сетью;
* поддержку программирования баз данных, включая: Oracle, Microsoft SQL Server, IBM, DB2, MySQL, SQLite, Sybase, PostgreSQL, и многое другое.

Qt – полностью объектно-ориентированная библиотека. Новая концепция ведения межобъектных коммуникаций, именуемая «сигналы и слоты», полностью заменяет былую, не вполне надежную модель обратных вызовов. Имеется также возможность обработки событий – например, нажатия клавиш клавиатуры, перемещения мыши и т. д.

Предоставляемая система расширений (plug-ins) позволяет создавать модули, расширяющие функциональные возможности создаваемых приложений. Эти расширения пользователи программы могут получать разных разработчиков.

Несмотря на то, что библиотека Qt изначально создавалась для языка программирования «C++», это вовсе не означает, что ее использование невозможно в других языках. Напротив, во многих языках программирования существуют модули для работы с этой библиотекой – например:

* Qt# в С#;
* PerlQt в Perl;
* PyQt в Python;
* РНР и т. д.

Программы, реализованные с помощью Qt, могут использовать язык сценариев QtScript. Эта технология позволяет пользователям приложения расширить возможности без изменения исходного кода и без перекомпоновки самого приложения изменить «поведение» приложения.

Qt прекрасно документирована, благодаря чему с помощью программы «QtAsisstant» всегда можно почерпнуть о ней любую интересующую информацию.

Если быть предельно кратким, то библиотеку Qt можно охарактеризовать в трех словах: простота, быстрота, мощность.

## Классы системы

Для придания принадлежности кода системы к конкретно данной системе, было создано пространство имен (namespace) «CGX». Это означает, что все классы, переменные, функции, интерфейсы и пр. будут иметь префикс пространства имен. Данный подход избавит дальнейшее расширение системы (плагины или доработка) от несанкционированного повреждения всех сущностей языка программирования.

Выделенные классы для реализации системы можно разделить на две группы:

* сущности (структуры данных);
* исполняющие механизмы.

Первый тип классов описывает типы данных, используемые в системе, например, тип «CGXProject». Все эти классы описывают сущности, а их функции являются лишь вспомогательным инструментарием интерфейсов. Второй тип классов манипулирует сущностями и выполняет какие-либо действия, например, запускает программу.

Так как язык Python имеет позднее связывание (динамическую типизацию), все значения параметров функций требуют проверку на соответствие типа.

### Сущности системы

Для реализации системы было выделено 8 сущностей:

Таблица 7.1 – Сущности системы

|  |  |  |
| --- | --- | --- |
| Сущность | Название класса | Описание |
| Проект | CGXProject | Данная сущность описывает проект, создаваемый менеджером отдела разработки. Далее атрибуты проекта дополняют начальники отделов. К проекту подключаются художники всех отделов |
| Метаинформация | CGXMetainfo | Данная сущность является описательной частью сущности «Проект». Она содержит информацию о заголовке, описании и сроков сдачи проекта |
| Дедлайн | CGXDeadline | Данная сущность хранит сроки создания и сдачи проекта |
| Заказчик | CGXCustomer | Данная сущность описывает заказчика проекта. |
| Контакты | CGXContacts | Данная сущность содержит данные по контактам заказчика: вебсайты, социальные сети, телефоны, мейлы, а так заметки |
| Сотрудник | CGXEmployee | Данная сущность описывает сотрудника студии. Поля «логин» и «пароль» заполняются не для каждого типа сотрудника (например, менеджер не имеет логина и пароля) |
| Программа | CGXProgram | Данная сущность описывает подключенные внешние программы. Были выделены важнейшие атрибуты, которые обеспечивают гибкость при работе с внешними приложениями |
| Переменная | CGXVariable | Данная сущность описывает переменную, которую пользователи будут использовать в различных аспектах системы |

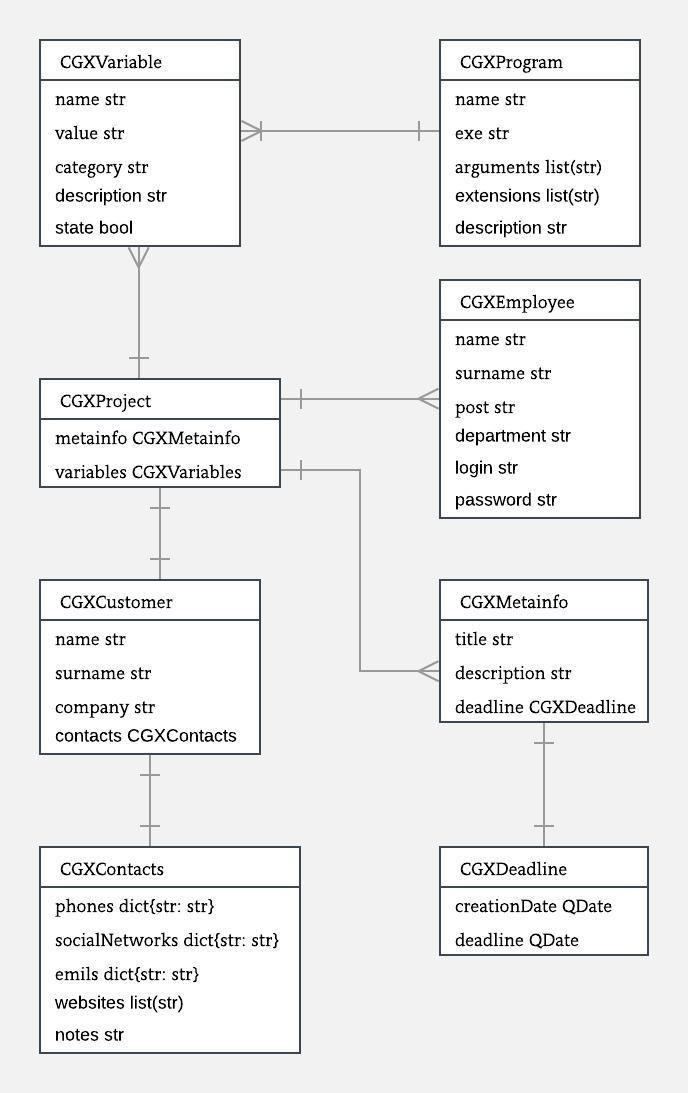
На рисунке 7.1 приведена схема логической структуры связей сущностей

Рис. 7.1 – Логическая структура связей сущностей

Далее присутствуют таблицы описания сущностей. В заголовке имя сущности.

|  |  |  |  |
| --- | --- | --- | --- |
| Контакты (CGXContacts) | | | |
| Атрибут | В классе | рy-тип | Описание |
| Телефоны | phones | dict{str: str} | Содержит пары { название : номер }. Данное поле имеет тип словарь и хранит все телефоны заказчика. Ключом является название телефона (например «Основной»), а значением – номер. |
| Соц. сети | socialNetworks | dict{str: str} | Содержит пары { название : ссылка }. Данное поле имеет тип словарь и хранит ссылки на страницы социальных сетей заказчика. Ключом является название соц. сети (например «Facebook»), а значением – ссылка. |
| Мессенджеры | messengers | dict{str: str} | Содержит пары { название : ссылка }. Данное поле имеет тип словарь и хранит ссылки на страницы месенджеров заказчика. Ключом является название месенджера (например, «Skype»), а значением – ссылка. |
| Электронная почта | emails | dict{str: str} | Содержит пары { название : ссылка }. Данное поле имеет тип словарь и хранит электронные почты заказчика. Ключом является название (например, «Основная»), а значением – почта. |
| Вебсайты | websites | list(str) | Данное поле имеет тип список и хранит ссылки на вебсайты заказчика. Ссылка хранится в строке. |
| Заметки | notes | str | HTML-cтрока, которая хранит заметки по заказчику |

|  |  |  |  |
| --- | --- | --- | --- |
| Дедлайн (CGXDeadline) | | | |
| Атрибут | В классе | рy-тип | Описание |
| Начало | creationDate | QDate | Хранит дату создания проекта |
| Срок сдачи | deadline | QDate | Хранит дату сдачи проекта |

|  |  |  |  |
| --- | --- | --- | --- |
| Метаинформация (CGXMetainfo) | | | |
| Атрибут | В классе | рy-тип | Описание |
| Заголовок | title | str | Данное поле хранит заголовок проекта |
| Описание | description | str | HTML-cтрока, которая хранит описание проекта |
| Сроки выполнения | deadline | CGXDeadline | Объект, который хранит сроки начала создания проекта и его дедлайн |

|  |  |  |  |
| --- | --- | --- | --- |
| Заказчик (CGXCustomer) | | | |
| Атрибут | В классе | рy-тип | Описание |
| Имя | name | str | Данное поле содержит имя заказчика |
| Фамилия | surname | str | Данное поле содержит фамилию заказчика |
| Компания | company | str | Данное поле содержит название организации заказчика |
| Контакты | contacts | CGXContacts | Данное поле хранит полную информацию о контактах заказчика |

|  |  |  |  |
| --- | --- | --- | --- |
| Сотрудник (CGXEmployee) | | | |
| Атрибут | В классе | рy-тип | Описание |
| Имя | name | str | Данное поле содержит имя сотрудника |
| Фамилия | surname | str | Данное поле содержит фамилию сотрудника |
| Логин | login | str | Данное поле содержит логин сотрудника (для подсистемы «CGX Artist» и «CGX Supervisor») |
| Пароль | password | str | Данное поле содержит пароль сотрудника (для подсистемы «CGX Artist» и «CGX Supervisor») |
| Должность | post | str | Содержит должность сотрудника |
| Отдел | department | str | Содержит отдел сотрудника (для подсистемы «CGX Artist» и «CGX Supervisor») |
| Переменная (CGXVariable) | | | |
| Атрибут | В классе | рy-тип | Описание |
| Название | name | str | Данное поле содержит название переменной |
| Значение | value | str | Данное поле содержит значение переменной |
| Состояние | state | bool | Данное поле является флагом активации переменной |
| Описание | description | str | Данное поле содержит описание переменной |
| Категория | category | str | Данное поле содержит категорию переменной (например, «Системная») |

|  |  |  |  |
| --- | --- | --- | --- |
| Программа (CGXProgram) | | | |
| Атрибут | В классе | рy-тип | Описание |
| Название | name | str | Данное поле содержит название программы |
| Исполн. файл | exe | str | Данное поле содержит полный путь до исполняемого файла программы |
| Расширения | extensions | list(str) | Данное поле имеет тип список и хранит расширения файлов, которые может запустить программа. Расширение хранится в виде строки. |
| Аргументы | arguments | list(str) | Данное поле имеет тип список и хранит аргументы запуска программы. Аргумент хранится в виде строки. |

|  |  |  |  |
| --- | --- | --- | --- |
| Проект (CGXProject) | | | |
| Атрибут | В классе | рy-тип | Описание |
| Метаинформация | metainfo | CGXMetainfo | Данный объект хранит всю метаинформацию о проекте |
| Переменные | variables | VariableContainer | Контейнер переменных проекта |

|  |  |  |  |
| --- | --- | --- | --- |
| Задача (CGXTask) | | | |
| Атрибут | В классе | рy-тип | Описание |
| Название | name | str | Данное поле хранит название задачи |
| Описание | description | str | HTML-cтрока, которая хранит описание задачи |
| Сроки выполнения | deadline | CGXDeadline | Объект, который хранит срок начала создания задачи и срок ее сдачи |

### Исполняющие механизмы

Поскольку различные аспекты системы должны настраиваться извне, было разработано гибкое решение данной задачи. В системе реализованы переменные, которые пользователи сами создают и инициализируют нужными значениями. Это очень схоже с переменными окружения операционной системы. Обработкой этих переменных, поиском ошибок, развертыванием занимается модуль под названием «VariableProcessing» (обработчик переменных). Этот модуль, содержит три класса:

Таблица 7.2 – Описание классов модуля «VariableProcessing»

|  |  |
| --- | --- |
| Класс | Описание |
| CGXVariableParser | Данное класс обеспечивает парсинг строк, содержащих переменные. Возможны 3 способа выделения переменных:   * внутренние переменные; * переменные системы; * все переменные.   Также можно проверить строку на наличие тех или иных видов переменных. |
| CGXRegexps | Данный класс является структурой, которая хранить шаблоны регулярных выражений для:   * системных переменных; * внутренних переменных; * все типы переменных.   Данная структура платформонезависимая, и возвращает правильные шаблоны для системных переменных. |
| CGXVariableProcessor | Данный класс является главным в модуле. Отвечает за манипулирование переменными в строках.  Позволяет:   * проверить переменную на наличие ошибок; "ScanError" * развернуть переменные в строке; "expandString, expandStringOS" * развернуть значение переменной; "expandVariable, expandVariableOS" * построить графики для строки; "stringToGraphs" * построить граф для переменной; "VariableToGraph"   Исключения:  NameError - если встречается недопустимое имя переменной  ReplicationError - если переменная содержит себя (или содержит переменные, которые содержат сами или переменные на уровне выше)  Функция «nameByError» позволяет получить имя переменной по коду ошибки. |

Все остальные исполняющие механизмы имеют типизированную структуру и не требуют рассмотрения. К ним относятся:

* модуль межсетевого взаимодействия (протокол TCP/IP);
* модуль запуска программ;
* модуль взаимодействия с базой данных (SQLite 3.0);

Модули связываются между собой с помощью сигналов и слотов, которые поддерживает фреймворк Qt.

### Вспомогательные классы

Для облегчения написания кода, а также для уменьшения его количества, были разработаны вспомогательные классы-обертки для работы с направленными графами, и операционной системой:

* DiGraphUtil;
* OSUtil.

Класс DigraphUtil обеспечивает полный функционал для работы с направленными графами. Он позволяет создавать граф, манипулировать им, исполнять алгоритмы поиска, а также выводить его структуру в текстовом или графическом виде. Данный класс используется в процессоре переменных (для глубокого парсинга строк, содержащих переменные) и при работе с древовидными структурами. Основан на библиотеке «xnetwork».

Класс OSUtil использует возможности класса DiGraphUtil для сохранения древовидной структуры на жесткий диск. Основными задачами данного класса стали: файловый ввод/вывод в разных форматах, контроль правильности имен директорий и файлов, работа с путями. Данный класс используется во многих частях системы. Основан на библиотеке «os».

### Графический интерфейс приложений

Фреймворк «Qt» позволяет реализовывать графические интерфейсы двумя методами:

* на используемом языке программирования за счет модуля «QtWidgets» (в нашем случае – «Python», а вообще – «С++»);
* на языке сценариев «QML».

Первый метод является более трудоемким в аспекте анимаций и лейаутинга, поэтому не будет рассматриваться.

Для реализации графических интерфейсов приложений системы был выбран второй метод – язык сценарием «QML».

QML – это описательный язык, он описывает, как выглядят и как взаимодействуют друг с другом элементы пользовательского интерфейса. В силу своего описательного характера этот язык не предполагает в себе использования конструкций программирования – например, циклов. И если требуется переместить с одного места на другое какой-либо объект, то этот процесс нужно описать. Впрочем, использовать те же циклы возможно благодаря встроенному в QML языку JavaScript. Для перехода к QML требуется небольшой переворот в сознании, чтобы понять философию создания приложений на QML.

QML – простой, легко осваиваемый и, в то же время, мощный язык программирования, который обладает элегантным синтаксисом. Он очень гибок, и рассчитан на создание пользовательских интерфейсов с использованием анимации. С его помощью можно создавать и воплощать собственные идеи и экспериментировать.

В QML встроен доступ ко всем имеющимся технологиям Qt – таким как, например, Qt/C++, QtwebKit, QtMobility, имеется также и доступ к метаинформации объектов – например, к свойствам, вызываемым методами (invokable methods), декларируемыми при помощи макроса Q\_INVOKABLE и т. п. QML обладает системой версионализации модулей.

Интегрированная среда разработки Qt Creator предоставляет хорошую поддержку этого языка.

Данный метод позволяет полностью отделить графический интерфейс от функциональной части приложения, что дает невероятную гибкость, как и при проектировании, так и при реализации.

# Описание контрольного примера использования системы

В данной главе будет описано использование разрабатываемой системы. На данный момент реализована лишь одна подсистема (неполностью) – «CGX Artist», которая является вспомогательным инструментом художника студии. Все изображения графического интерфейса приведены в приложении Ж.

## Подсистема «CGX Artist»

Главное окно данного приложения состоит из 3 основных частей:

* Верхняя панель (1, отмечено красным) ;
* Панель представлений (2, отмечено фиолетовым);
* Представление (3, отмеченно зеленым).



Рис. 8.1 – Структура главного окна

### Верхняя панель

На верхней панели располагаются кнопки отображения текущего проекта (раздел 3), текущей задачи (раздел 2) и атрибуты текущего пользователя (раздел 1).

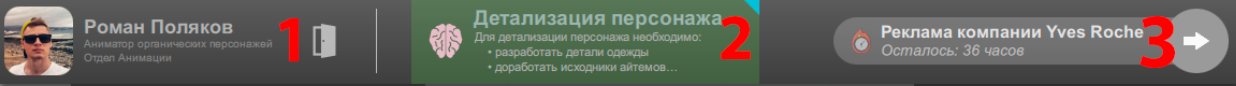


Рис. 8.2 – Структура верхней панели

Таблица 8.1 – Информация верхней панели

|  |  |  |
| --- | --- | --- |
| Раздел | Номер раздела | Описание отображаемой информации |
| Атрибуты пользователя | 1 | Имя и фамилия художника;  Отдел, в котором работает художник;  Должность художника |
| Текущая задача | 2 | Название текущей задачи;  Краткое описание задачи |
| Текущий проект | 3 | Название проекта;  Оставшееся время до сдачи |

Таблица 8.2 – Функции верхней панели

|  |  |
| --- | --- |
| Действие | Описание |
| Выход из приложения | Для выхода из приложения необходимо нажать на кнопку с иконкой двери в разделе 1 |
| Смена текщей задачи | Для смены текущей задачи необходимо нажать на зеленую кнопку в разделе 2 |
| Смена текущего проекта | Для смены текущего проекта необходимо нажать на серую кнопку с белой стрелкой в разделе 3 |

### Панель представлений

На данной панели располагаются кнопки для смены представлений и кнопка запуска выбранной программы.

Есть 4 доступных представления главного окна:

* Представление структуры проекта (1я кнопка);
* Представление управления внешними программами (2я кнопка);
* Представление управления переменными (3я кнопка);
* Представление настроек приложения «CGX Artist»;

Реализовано лишь представление структуры проекта, поэтому в следующих разделах данной главы будет рассмотренно лишь оно.

### Форма представления проекта

На форме панели располагаются: браузер файловой структуры проекта (раздел 1, красный), обозреватель содержимого определенныой директории (раздел 2, зеленый), окно для отправки файлов и просмотра переменных проекта.

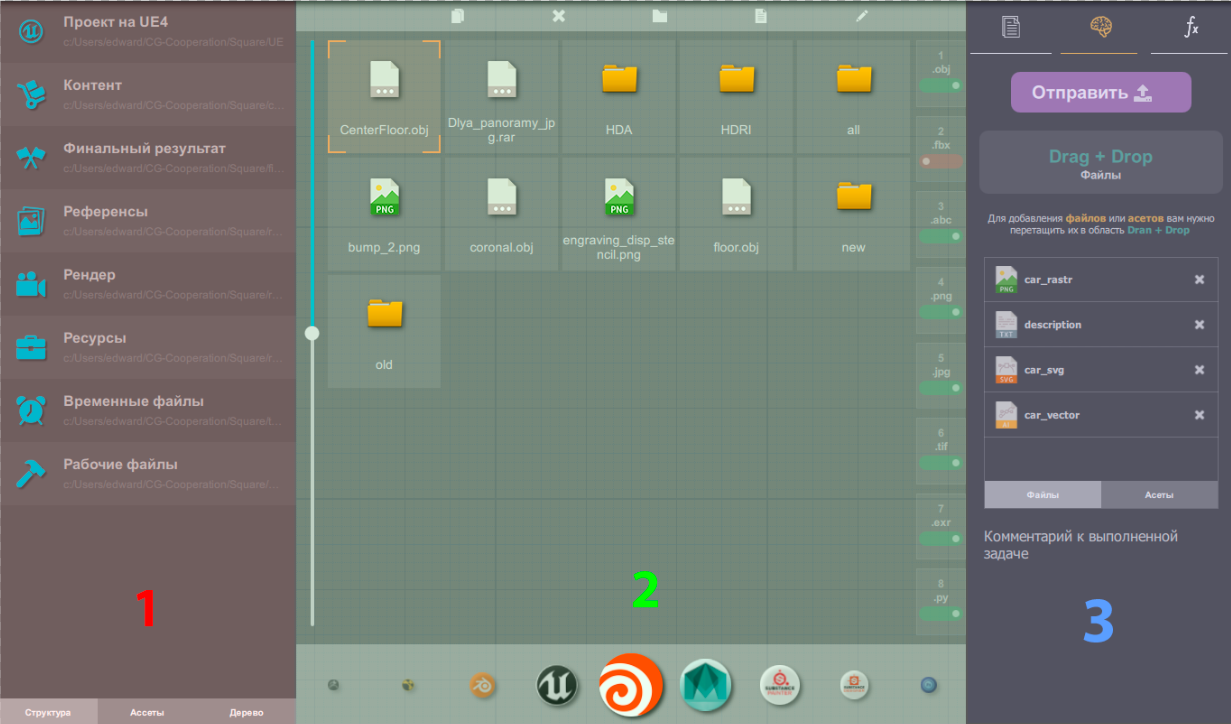


Рис. 8.3 – Структура представления проекта

Браузер (раздел 1) дает возможность обращаться к файлам проекта в трех видах:

* **Структура** – отображает шаблонные имена (и иконки) директорий проекта. Показывает лишь верхнюю часть проекта.
* **Дерево** – отображает файловую иерархию проекта.
* **Ассеты** – отображает список ассетов проекта.

Для смены вида необходимо нажать (тем самым активировав) на одну из трёх кнопок внизу формы



Рис. 8.4 – Виды представления структуры проекта

Внизу нижней части каждого вида (кроме «Структура») располагаются кнопки для добавления, удаления и копирования файлов, папок или ассетов.

Все три вида представления связаны с формой отображения содержимого конкретной директории (браузер содержимого директории). При выделении Директории или ассета в разделе 1, браузер содержимого сразу же отображает весь контент выбранного эелемента.

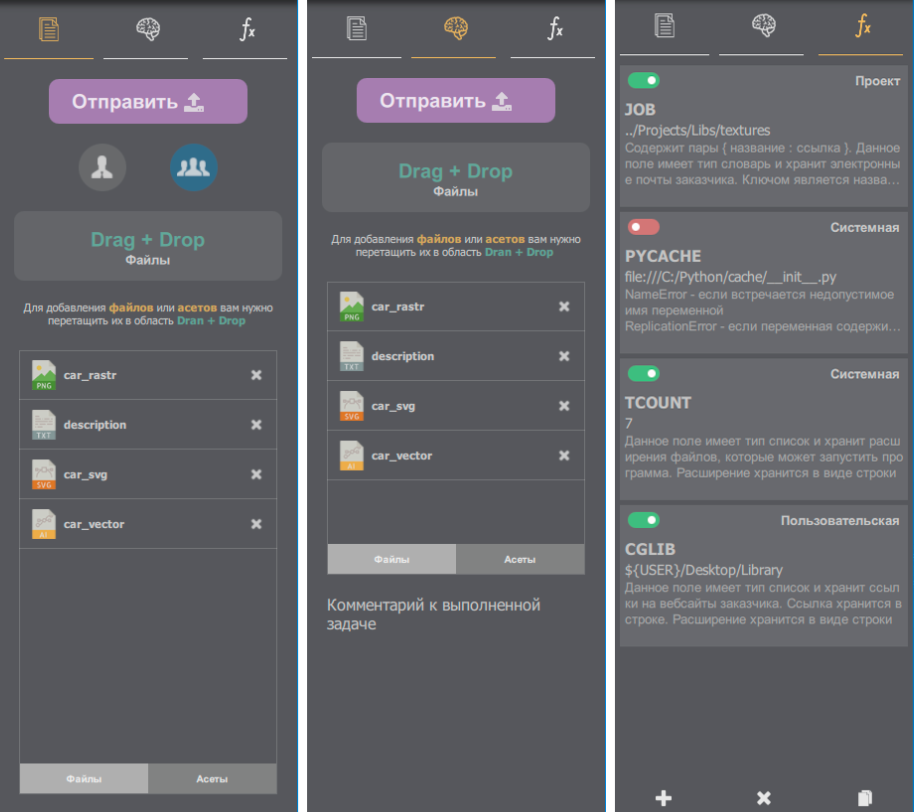


Рис. 8.5 – Структура обозревателя директорий

Данное окно состоит из 5 частей. Обозреватель – часть 5 (синяя), инструменты – часть 1 (красная), слайдер для масштабирования – часть 2 (голубая), инструменты фильтрации и сортировки – часть 3 (желтая) и список программ, способный открыть выбранный файл – часть 4 (красная).

Элементы фильтрации настраиваются пользователем в представлении настроект приложения (не реализованная часть). Каждый жлемент состоит из типа файла, номера сортировки и флага, который позволяет ключить или выключить отображение файлов определенного типа. Инструменты в верхней части окна позволяют добавлять, удалять, переименовывать и копировать файлы и папки. При выделении приложения в нижней части окна, можно запустить его, нажав на зеленную кнопку на панели представлений.

Окно отправки файлов и просмотра переменных проекта состоит из трез форм: первое – для отправки файлов начальнику или коллегам, второе – для отправки отчета по задачам, третье – для быстрого просмотра переменных проекта. Возможна отправка ассетов. Для добавления файлов (ассетов) необходимо перетащить (функционал drag-drop) из браузера проекта (или дириктории) в область «Dran + Drop».



**Заключение**

В результате выполнения выпускной квалификационной работы, был проведен анализ работы студии, что позволило выявить недостатки организации рабочего процесса ее сотрудников.

Обоснована целесообразность разработки собственной системы, так как цена аналогичного продукта во много раз превышает затраты на разработку. Выполнено техническое проектирование программного продукта и описание реализации некоторых его частей.

Система осталась частично реализованной, поэтому невозможно сказать, что обладает всем необходимым функционалом для организации производственного процесса художников, начальников и менеджера студии на всех этапах выполнения проекта.

**Список используемых источников**

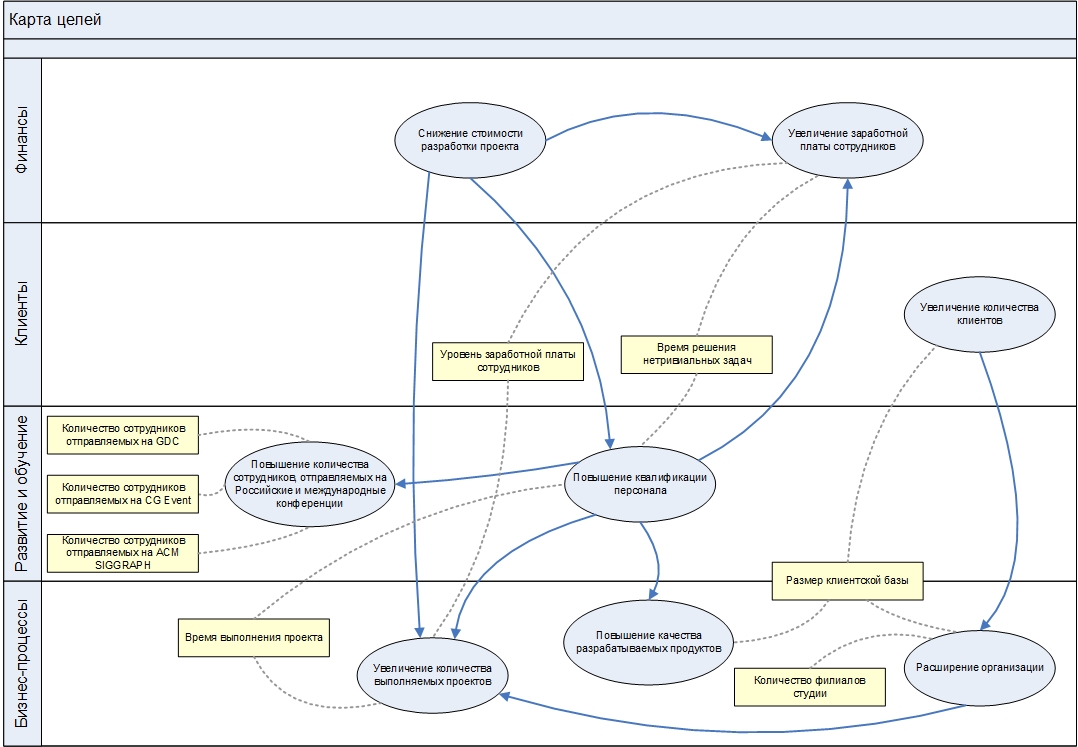
1. Рудинский И.Д. Архитектура и разработка корпоративных информационных систем [Текст]: учеб. - метод. пособие по вып. курс. проекта для студ., обуч. в бакалавриате по напр. подгот. "Прикладная информатика" (профиль подгот. "Прикладная информатика в экономике") / И. Д. Рудинский, 2015. - 77 с.
2. ГОСТ 34.601-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Заголовок: Автоматизированные системы стадии создания. — М.: Изд-во стандартов, 1990.
3. ГОСТ 34.201–89. Информационная технология. Комплекс стандартов на автоматизированные системы. Заголовок: Виды, комплектность и обозначения документов при создании автоматизированных систем — М.: Изд-во стандартов, 1990.
4. Рудинский И.Д. Архитектура и разработка корпоративных информационных систем. Учебно-методическое пособие по выполнению курсового проекта для студентов, обучающихся в бакалавриате по направлению подготовки «Прикладная информатика» (профиль подготовки – «Прикладная информатика в экономике») – Калининград: КГТУ, 2015.
5. Методические указания к лабораторным работам по дисциплине "Методы и средства проектирования информационных систем" по специальности 230102.65 – Автоматизированные системы обработки информации и управления. Калининград 2008-85 м. п.л.
6. Сайт студии «Глюкоза продакшн» - <http://www.glukoza-production.ru/>
7. Сайт студии «CGF» - <http://www.cgfww.com/about>
8. Документация системы бизнесс-моделирования «Business Studio» - <http://www.businessstudio.ru/wiki/docs/v4/doku.php/start>
9. Справочная система фреймворка «Qt» - https://doc.qt.io/qt-5/
10. Бизнес-модель студий

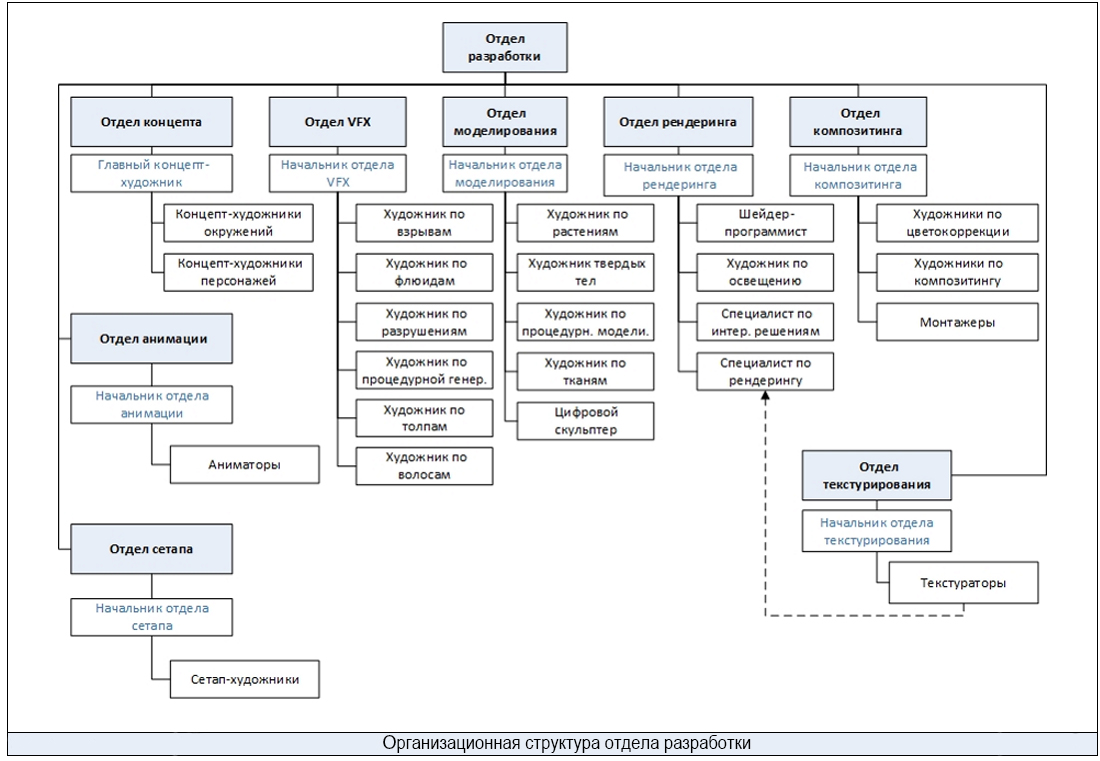
**Бизнес-модель студий**

titletitletitle

title

titletitle





1. Описание бизнес-процессов студии

**Описание бизнес-процессов**

**студии**

Термины и сокращения, используемые при описании:

|  |  |  |
| --- | --- | --- |
| {Граница} | – | внешний поставщик или потребитель стрелки. Находится за рамками моделируемой системы. Название не детализируется в случаях однозначного понимания читателями диаграммы или в случаях неоднозначности. |
| {Туннель} | – | поставщик или потребитель стрелки. Название не детализируется в случаях однозначного понимания читателями диаграммы. |
| Бизнес-процесс | – | последовательность действий (подпроцессов), направленная на получение заданного результата, ценного для организации (далее Процесс). |
| Владелец процесса | – | должностное лицо, несущее ответственность за получение результата процесса и обладающее полномочиями для распоряжения ресурсами, необходимыми для выполнения процесса. |
| Входы бизнес-процесса | – | ресурсы (материальные, информационные), необходимые для выполнения и получения результата процесса, которые потребляются или преобразовываются при выполнении процесса. |
| Выходы бизнес-процесса | – | объекты (материальные или информационные), являющиеся результатом выполнения бизнес-процесса, потребляемые другими бизнес-процессами или внешними по отношению к организации клиентами. |
| Исполнитель процесса | – | подразделение или должность сотрудника, выполняющего процесс. |
| Механизмы бизнес- процесса | – | ресурсы (технологические, трудовые), используемые для выполнения процесса, целиком не потребляющиеся при выполнении одной итерации процесса. |
| Организационная структура управления | – | совокупность специализированных функциональных подразделений, взаимосвязанных в процессе обоснования, выработки, принятия и реализации управленческих решений (далее Организационная структура). |
| Подпроцесс | – | бизнес-процесс, являющийся составной частью вышестоящего процесса. |
| Процедура | – | бизнес-процесс нижнего уровня, содержащий последовательность конечных (не требующих дополнительной детализации) операций (функций). |
| Управление бизнес- процесса | – | управляющие воздействия, регламентирующие выполнение процесса. |

А0 Создание графического контента

Содержание деятельности

Содержанием деятельности по процессу «А0 Создание графического контента» является разработка (по требованию заказчика) комплекса 3D-моделей, текстур, анимаций и симуляций, скомпонованных в единый видеоролик или же статическое изображение для применения в рекламе, кинофильме, мобильном приложении или видео игре.

Владелец процесса

Владельцем процесса является:

* Директор

Исполнители процесса

Исполнителями процесса являются:

* Менеджер
* Отдел разработки

Начало выполнения процесса

Начало выполнения процесса определяется в общем ТЗ.

Результат процесса

Основным результатом процесса является графический контент, соответствующий общему ТЗ и удовлетворяющий заказчика.

Требования к срокам выполнения процесса

Определяются в общем ТЗ

Документация процесса

Выполнение процесса «Создание графического контента» регламентируется следующей плановой или нормативно-методической документацией:

|  |
| --- |
| * Список активных проектов |

ВЗАИМОДЕЙСТВИЕ С ДРУГИМИ ПРОЦЕССАМИ И ВНЕШНЕЙ СРЕДОЙ

При выполнении процесса «А0 Создание графического контента» осуществляется взаимодействие с другими процессами и с внешней средой по входу, выходу, управлению и механизмам:

Взаимодействие по Входу

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Заказчик |  |  | {Граница} |

Взаимодействие по Выходу

| № | Выход | Объекты | Передается | |
| --- | --- | --- | --- | --- |
| Получатель | Процесс |
| 1. | Графический контент | Реализованный графический контент |  | {Граница} |

Управление процесса

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Загруженность студии | Список активных проектов |  | {Граница} |

Механизмы процесса

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Отдел разработки |  |  | {Туннель} |
| 2. | Сотрудники студии |  |  | {Граница} |

ОРГАНИЗАЦИЯ ВЫПОЛНЕНИЯ ПРОЦЕССА

А1 Формирование общего ТЗ

Содержанием деятельности по процессу «А1 Формирование общего ТЗ» является собеседование с заказчиком, в ходе которого начальники отделов и менеджер составляют общее техническое задание.

Владелец подпроцесса

* Менеджер

Исполнители подпроцесса

* Главный концепт-художник (Отдел концепта)
* Начальник отдела VFX (Отдел VFX)
* Начальник отдела анимации (Отдел анимации)
* Начальник отдела композитинга (Отдел композитинга)
* Начальник отдела моделирования (Отдел моделирования)
* Начальник отдела рендеринга (Отдел рендеринга)
* Начальник отдела сетапа (Отдел сетапа)
* Начальник отдела текстурирования (Отдел текстурирования)

Начало выполнения

Начало выполнения процесса после подписания договора на выполнение проекта.

Требования к срокам

Требования к срокам не более 2 часов

Результат выполнения

Результатом выполнения процесса является общее ТЗ.

Документация подпроцесса

Выполнение подпроцесса регламентируется следующей плановой и нормативно-методической документацией:

|  |
| --- |
| * Список активных проектов |

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Заказчик |  |  | {Граница} |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | ОТЗ | Общее ТЗ | Начальники | А2.1 Постановка задач |
| Начальники | А2.3 Формирование локальных ТЗ |

Управление подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Загруженность студии | Список активных проектов |  | {Граница} |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Менеджер |  |  | {Граница} |
| 2. | Начальники отделов |  |  | {Граница} |

А2 Постановка задач и формирование локальных ТЗ

Содержанием деятельности по процессу «А2 Постановка задач и формирование локальных ТЗ» является постановка и планирование задач по проекту, а также составление ТЗ для каждого отдела разработки.

Владелец подпроцесса

* Менеджер

Исполнители подпроцесса

* Главный концепт-художник (Отдел концепта)
* Начальник отдела VFX (Отдел VFX)
* Начальник отдела анимации (Отдел анимации)
* Начальник отдела композитинга (Отдел композитинга)
* Начальник отдела моделирования (Отдел моделирования)
* Начальник отдела рендеринга (Отдел рендеринга)
* Начальник отдела сетапа (Отдел сетапа)
* Начальник отдела текстурирования (Отдел текстурирования)

Начало выполнения

Начало выполнения процесса сразу после формирования общего ТЗ.

Требования к срокам

Требования к срокам не более 1 часа

Результат выполнения

Результатом выполнения процесса является список задач и локальные ТЗ.

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | ОТЗ | Общее ТЗ | Начальники | А1.5 Установка сроков сдачи |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Задачи | Список задач | Концепт-художники окружений  Концепт-художники персонажей | А3.1 Создание концепта |
| Аниматоры  Сетап-художники  Текстураторы  Художник по волосам  Художник по тканям  Художник твердых тел  Цифровой скульптер | А3.2 Разработка персонажей |
| Текстураторы  Художник по процедурной генерации  Художник по процедурному моделированию  Художник по растениям  Художник твердых тел | А3.3 Разработка окружений |
| Художник по процедурной генерации  Художник по разрушениям  Художник по толпам  Художник по флюидам | А3.4 Создание визуальных эффектов |
| Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |
| 2. | Локальные ТЗ | ТЗ на концепты | Концепт-художники окружений  Концепт-художники персонажей | А3.1 Создание концепта |
| ТЗ на персонажей | Аниматоры  Сетап-художники  Текстураторы  Художник по волосам  Художник по тканям  Художник твердых тел  Цифровой скульптер | А3.2 Разработка персонажей |
| ТЗ на окружение | Текстураторы  Художник по процедурной генерации  Художник по процедурному моделированию  Художник по растениям  Художник твердых тел | А3.3 Разработка окружений |
| ТЗ на визуальные эффекты | Художник по процедурной генерации  Художник по разрушениям  Художник по толпам  Художник по флюидам | А3.4 Создание визуальных эффектов |
| ТЗ на визуализацию | Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Начальники отделов |  |  | {Граница} |

А3 Реализация проекта

Содержанием деятельности по процессу «А3 Реализация проекта» является выполнение работ, связанных с реализацией проекта. Создаются 3D-модели, текстуры и анимации. Все это собирается вместе и визуализируется.

Исполнители подпроцесса

* Отдел разработки

Начало выполнения

Начало выполнения процесса определяется в общем ТЗ.

Требования к срокам

Требования к срокам определяется в общем ТЗ

Результат выполнения

Результатом выполнения процесса является графический контент, тематика которого определяется в общем ТЗ.

Документация подпроцесса

Выполнение подпроцесса регламентируется следующей плановой и нормативно-методической документацией:

|  |
| --- |
| * ТЗ на визуализацию |
| * ТЗ на визуальные эффекты |
| * ТЗ на концепты |
| * ТЗ на окружение |
| * ТЗ на персонажей |

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Задачи | Список задач | Начальники | А2.2 Планирование задач |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Графический контент | Реализованный графический контент |  | {Граница} |
| Менеджер | А4.1 Отправка заказчику |

Управление подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Локальные ТЗ | ТЗ на визуализацию  ТЗ на визуальные эффекты  ТЗ на концепты  ТЗ на окружение  ТЗ на персонажей | Начальники | А2.3 Формирование локальных ТЗ |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Отдел разработки |  |  | {Граница} |

А4 Отправка заказчику и доработка

Содержанием деятельности по процессу «А4 Отправка заказчику и доработка» является отправка заказчику реализованного проекта и выполнение доработки на основе списка правок.

Исполнители подпроцесса

* Менеджер
* Отдел разработки

Начало выполнения

Начало выполнения процесса после реализации проекта.

Требования к срокам

Требования к срокам не более 1 дня

Результат выполнения

Результатом выполнения процесса является финальный продукт.

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Графический контент | Реализованный графический контент | Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Менеджер |  |  | {Граница} |
| 2. | Отдел разработки |  |  | {Туннель} |

СОСТАВ НАБОРОВ ОБЪЕКТОВ И РОЛЕЙ, УЧАСТВУЮЩИХ В ВЫПОЛНЕНИИ ПРОЦЕССА

| № | Набор объектов | Объекты, входящие в состав набора |
| --- | --- | --- |
| 1. | Реализованный графический контент |  |

| № | Роль | Субъект | Подразделение | Предмет деятельности |
| --- | --- | --- | --- | --- |
| 1. | Начальники | Главный концепт-художник | Отдел концепта |  |
| Начальник отдела VFX | Отдел VFX |  |
| Начальник отдела анимации | Отдел анимации |  |
| Начальник отдела композитинга | Отдел композитинга |  |
| Начальник отдела моделирования | Отдел моделирования |  |
| Начальник отдела рендеринга | Отдел рендеринга |  |
| Начальник отдела сетапа | Отдел сетапа |  |
| Начальник отдела текстурирования | Отдел текстурирования |  |

А0 Создание графического контента

Содержание деятельности

Содержанием деятельности по процессу «А1 Формирование общего ТЗ» является собеседование с заказчиком, в ходе которого начальники отделов и менеджер составляют общее техническое задание.

Владелец процесса

Владельцем процесса является:

* Менеджер

Исполнители процесса

Исполнителями процесса являются:

* Главный концепт-художник (Отдел концепта)
* Начальник отдела VFX (Отдел VFX)
* Начальник отдела анимации (Отдел анимации)
* Начальник отдела композитинга (Отдел композитинга)
* Начальник отдела моделирования (Отдел моделирования)
* Начальник отдела рендеринга (Отдел рендеринга)
* Начальник отдела сетапа (Отдел сетапа)
* Начальник отдела текстурирования (Отдел текстурирования)

Начало выполнения процесса

Начало выполнения процесса после подписания договора на выполнение проекта.

Результат процесса

Основным результатом процесса является общее ТЗ.

Требования к срокам выполнения процесса

не более 2 часов

Документация процесса

|  |
| --- |
| * Список активных проектов |

ВЗАИМОДЕЙСТВИЕ С ДРУГИМИ ПРОЦЕССАМИ И ВНЕШНЕЙ СРЕДОЙ

При выполнении процесса «А1 Формирование общего ТЗ» осуществляется взаимодействие с другими процессами и с внешней средой по входу, выходу, управлению и механизмам:

Взаимодействие по Входу

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Заказчик |  |  | {Граница} |

Взаимодействие по Выходу

| № | Выход | Объекты | Передается | |
| --- | --- | --- | --- | --- |
| Получатель | Процесс |
| 1. | ОТЗ | Общее ТЗ | Начальники | А2.1 Постановка задач |
| Начальники | А2.3 Формирование локальных ТЗ |

Управление процесса

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Загруженность студии | Список активных проектов |  | {Граница} |

Механизмы процесса

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Менеджер |  |  | {Граница} |
| 2. | Начальники отделов |  |  | {Граница} |

ОРГАНИЗАЦИЯ ВЫПОЛНЕНИЯ ПРОЦЕССА

А1.1 Определение тематики и стилистики проекта

Содержанием деятельности по процессу «А1.1 Определение тематики и стилистики проекта» является этап собеседования, в результате которого выявляется тема и стиль проекта.

Владелец подпроцесса

* Менеджер

Исполнители подпроцесса

* Главный концепт-художник (Отдел концепта)

Начало выполнения

Начало выполнения процесса по желанию заказчика, но не ранее подписания договора о выполнении проекта.

Требования к срокам

Требования к срокам не более 20 минут

Результат выполнения

Результатом выполнения процесса является тема и стиль проекта.

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Заказчик |  |  | {Граница} |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Обновленное ОТЗ | Общее ТЗ | Начальник отдела моделирования  Начальник отдела текстурирования | А1.2 Определение информации о персонажах и окружениях |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Главный концепт-художник |  |  | {Граница} |

А1.2 Определение информации о персонажах и окружениях

Содержанием деятельности по процессу «А1.2 Определение информации о персонажах и окружениях» является этап собеседования, в результате которого выявляется информация о количестве персонажей и окружений, а также их детализации.

Владелец подпроцесса

* Менеджер

Исполнители подпроцесса

* Начальник отдела моделирования (Отдел моделирования)
* Начальник отдела текстурирования (Отдел текстурирования)

Начало выполнения

Начало выполнения процесса сразу после определения тематики.

Требования к срокам

Требования к срокам не более 30 минут

Результат выполнения

Результатом выполнения процесса является полная информация о сложности персонажей и окружений.

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Заказчик |  |  | {Граница} |
| 2. | Обновленное ОТЗ | Общее ТЗ | Главный концепт-художник | А1.1 Определение тематики и стилистики проекта |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Обновленное ОТЗ | Общее ТЗ | Начальник отдела VFX | А1.3 Определение информации о визуальных эффектах |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Начальник отдела VFX |  |  | {Граница} |
| 2. | Начальник отдела анимации |  |  | {Граница} |
| 3. | Начальник отдела моделирования |  |  | {Граница} |
| 4. | Начальник отдела сетапа |  |  | {Граница} |
| 5. | Начальник отдела текстурирования |  |  | {Граница} |

А1.3 Определение информации о визуальных эффектах

Содержанием деятельности по процессу «А1.3 Определение информации о визуальных эффектах» является этап собеседования, в результате которого выявляется информация о визуальных эффектах в проекте.

Владелец подпроцесса

* Менеджер

Исполнители подпроцесса

* Начальник отдела VFX (Отдел VFX)

Начало выполнения

Начало выполнения процесса сразу после определения информации о персонажах и окружениях.

Требования к срокам

Требования к срокам не более 30 минут

Результат выполнения

Результатом выполнения процесса является полная информация о сложности визуальных эффектов.

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Заказчик |  |  | {Граница} |
| 2. | Обновленное ОТЗ | Общее ТЗ | Начальник отдела моделирования  Начальник отдела текстурирования | А1.2 Определение информации о персонажах и окружениях |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Обновленное ОТЗ | Общее ТЗ | Менеджер  Начальник отдела рендеринга | А1.4 Определение информации о выходном формате |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Начальник отдела VFX |  |  | {Граница} |

А1.4 Определение информации о выходном формате

Содержанием деятельности по процессу «А1.4 Определение информации о выходном формате» является этап собеседования, в результате которого выявляется информация о формате готового результата.

Исполнители подпроцесса

* Менеджер
* Начальник отдела рендеринга (Отдел рендеринга)

Начало выполнения

Начало выполнения процесса сразу после определения информации о визуальных эффектах.

Требования к срокам

Требования к срокам не более 5 минут

Результат выполнения

Результатом выполнения процесса является формат выходных данных.

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Заказчик |  |  | {Граница} |
| 2. | Обновленное ОТЗ | Общее ТЗ | Начальник отдела VFX | А1.3 Определение информации о визуальных эффектах |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Обновленное ОТЗ | Общее ТЗ | Начальники | А1.5 Установка сроков сдачи |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Начальник отдела композитинга |  |  | {Граница} |
| 2. | Начальник отдела рендеринга. |  |  | {Граница} |

А1.5 Установка сроков сдачи

Содержанием деятельности по процессу «А1.5 Установка сроков сдачи» является этап собеседования, в результате которого производится установка сроков сдачи проекта.

Владелец подпроцесса

* Менеджер

Исполнители подпроцесса

* Главный концепт-художник (Отдел концепта)
* Начальник отдела VFX (Отдел VFX)
* Начальник отдела анимации (Отдел анимации)
* Начальник отдела композитинга (Отдел композитинга)
* Начальник отдела моделирования (Отдел моделирования)
* Начальник отдела рендеринга (Отдел рендеринга)
* Начальник отдела сетапа (Отдел сетапа)
* Начальник отдела текстурирования (Отдел текстурирования)

Начало выполнения

Начало выполнения процесса сразу после определения сведений о формате готового результата.

Требования к срокам

Требования к срокам не более 5 минут

Результат выполнения

Результатом выполнения процесса является сроки сдачи проекта.

Документация подпроцесса

Выполнение подпроцесса регламентируется следующей плановой и нормативно-методической документацией:

|  |
| --- |
| * Список активных проектов |

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Заказчик |  |  | {Граница} |
| 2. | Обновленное ОТЗ | Общее ТЗ | Менеджер  Начальник отдела рендеринга | А1.4 Определение информации о выходном формате |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | ОТЗ | Общее ТЗ | Начальники | А2.1 Постановка задач |
| Начальники | А2.3 Формирование локальных ТЗ |

Управление подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Загруженность студии | Список активных проектов |  | {Граница} |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Менеджер |  |  | {Граница} |

| № | Роль | Субъект | Подразделение | Предмет деятельности |
| --- | --- | --- | --- | --- |
| 1. | Начальники | Главный концепт-художник | Отдел концепта |  |
| Начальник отдела VFX | Отдел VFX |  |
| Начальник отдела анимации | Отдел анимации |  |
| Начальник отдела композитинга | Отдел композитинга |  |
| Начальник отдела моделирования | Отдел моделирования |  |
| Начальник отдела рендеринга | Отдел рендеринга |  |
| Начальник отдела сетапа | Отдел сетапа |  |
| Начальник отдела текстурирования | Отдел текстурирования |  |

А2 Постановка задач и формирование локальных ТЗ

Содержание деятельности

Содержанием деятельности по процессу «А2 Постановка задач и формирование локальных ТЗ» является постановка и планирование задач по проекту, а также составление ТЗ для каждого отдела разработки.

Владелец процесса

Владельцем процесса является:

* Менеджер

Исполнители процесса

Исполнителями процесса являются:

* Главный концепт-художник (Отдел концепта)
* Начальник отдела VFX (Отдел VFX)
* Начальник отдела анимации (Отдел анимации)
* Начальник отдела композитинга (Отдел композитинга)
* Начальник отдела моделирования (Отдел моделирования)
* Начальник отдела рендеринга (Отдел рендеринга)
* Начальник отдела сетапа (Отдел сетапа)
* Начальник отдела текстурирования (Отдел текстурирования)

Начало выполнения процесса

Начало выполнения процесса сразу после формирования общего ТЗ.

Результат процесса

Основным результатом процесса является список задач и локальные ТЗ.

Требования к срокам выполнения процесса

не более 1 часа

ВЗАИМОДЕЙСТВИЕ С ДРУГИМИ ПРОЦЕССАМИ И ВНЕШНЕЙ СРЕДОЙ

При выполнении процесса «А2 Постановка задач и формирование локальных ТЗ» осуществляется взаимодействие с другими процессами и с внешней средой по входу, выходу, управлению и механизмам:

Взаимодействие по Входу

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | ОТЗ | Общее ТЗ | Начальники | А1.5 Установка сроков сдачи |

Взаимодействие по Выходу

| № | Выход | Объекты | Передается | |
| --- | --- | --- | --- | --- |
| Получатель | Процесс |
| 1. | Задачи | Список задач | Концепт-художники окружений  Концепт-художники персонажей | А3.1 Создание концепта |
| Аниматоры  Сетап-художники  Текстураторы  Художник по волосам  Художник по тканям  Художник твердых тел  Цифровой скульптер | А3.2 Разработка персонажей |
| Текстураторы  Художник по процедурной генерации  Художник по процедурному моделированию  Художник по растениям  Художник твердых тел | А3.3 Разработка окружений |
| Художник по процедурной генерации  Художник по разрушениям  Художник по толпам  Художник по флюидам | А3.4 Создание визуальных эффектов |
| Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |
| 2. | Локальные ТЗ | ТЗ на концепты | Концепт-художники окружений  Концепт-художники персонажей | А3.1 Создание концепта |
| ТЗ на персонажей | Аниматоры  Сетап-художники  Текстураторы  Художник по волосам  Художник по тканям  Художник твердых тел  Цифровой скульптер | А3.2 Разработка персонажей |
| ТЗ на окружение | Текстураторы  Художник по процедурной генерации  Художник по процедурному моделированию  Художник по растениям  Художник твердых тел | А3.3 Разработка окружений |
| ТЗ на визуальные эффекты | Художник по процедурной генерации  Художник по разрушениям  Художник по толпам  Художник по флюидам | А3.4 Создание визуальных эффектов |
| ТЗ на визуализацию | Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |

Механизмы процесса

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Начальники отделов |  |  | {Граница} |

ОРГАНИЗАЦИЯ ВЫПОЛНЕНИЯ ПРОЦЕССА

А2.1 Постановка задач

Содержанием деятельности по процессу «А2.1 Постановка задач» является постановка заданий начальниками отделов для своих подчиненных.

Владелец подпроцесса

* Менеджер

Исполнители подпроцесса

* Главный концепт-художник (Отдел концепта)
* Начальник отдела VFX (Отдел VFX)
* Начальник отдела анимации (Отдел анимации)
* Начальник отдела композитинга (Отдел композитинга)
* Начальник отдела моделирования (Отдел моделирования)
* Начальник отдела рендеринга (Отдел рендеринга)
* Начальник отдела сетапа (Отдел сетапа)
* Начальник отдела текстурирования (Отдел текстурирования)

Начало выполнения

Начало выполнения процесса сразу после формирования общего ТЗ.

Требования к срокам

Требования к срокам не более 30 минут

Результат выполнения

Результатом выполнения процесса является формулировка задач.

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | ОТЗ | Общее ТЗ | Начальники | А1.5 Установка сроков сдачи |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Формулировка задач | Список задач | Начальники | А2.2 Планирование задач |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Начальники отделов |  |  | {Граница} |

А2.2 Планирование задач

Содержанием деятельности по процессу «А2.2 Планирование задач» является планирование поставленных задач (определение сроков для каждой задачи).

Владелец подпроцесса

* Главный концепт-художник (Отдел концепта)

* Начальник отдела VFX (Отдел VFX)

* Начальник отдела анимации (Отдел анимации)

* Начальник отдела композитинга (Отдел композитинга)

* Начальник отдела моделирования (Отдел моделирования)

* Начальник отдела рендеринга (Отдел рендеринга)

* Начальник отдела сетапа (Отдел сетапа)

* Начальник отдела текстурирования (Отдел текстурирования)

Исполнители подпроцесса

* Главный концепт-художник (Отдел концепта)
* Начальник отдела VFX (Отдел VFX)
* Начальник отдела анимации (Отдел анимации)
* Начальник отдела композитинга (Отдел композитинга)
* Начальник отдела моделирования (Отдел моделирования)
* Начальник отдела рендеринга (Отдел рендеринга)
* Начальник отдела сетапа (Отдел сетапа)
* Начальник отдела текстурирования (Отдел текстурирования)

Начало выполнения

Начало выполнения процесса сразу после постановки задач.

Требования к срокам

Требования к срокам не более 10 минут

Результат выполнения

Результатом выполнения процесса является готовые задачи.

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Формулировка задач | Список задач | Начальники | А2.1 Постановка задач |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Задачи | Список задач | Начальники | А2.3 Формирование локальных ТЗ |
| Концепт-художники окружений  Концепт-художники персонажей | А3.1 Создание концепта |
| Аниматоры  Сетап-художники  Текстураторы  Художник по волосам  Художник по тканям  Художник твердых тел  Цифровой скульптер | А3.2 Разработка персонажей |
| Текстураторы  Художник по процедурной генерации  Художник по процедурному моделированию  Художник по растениям  Художник твердых тел | А3.3 Разработка окружений |
| Художник по процедурной генерации  Художник по разрушениям  Художник по толпам  Художник по флюидам | А3.4 Создание визуальных эффектов |
| Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Начальники отделов |  |  | {Граница} |

А2.3 Формирование локальных ТЗ

Содержанием деятельности по процессу «А2.3 Формирование локальных ТЗ» является формирование ТЗ начальниками отделов для своих подчиненных.

Владелец подпроцесса

* Главный концепт-художник (Отдел концепта)

* Начальник отдела VFX (Отдел VFX)

* Начальник отдела анимации (Отдел анимации)

* Начальник отдела композитинга (Отдел композитинга)

* Начальник отдела моделирования (Отдел моделирования)

* Начальник отдела рендеринга (Отдел рендеринга)

* Начальник отдела сетапа (Отдел сетапа)

* Начальник отдела текстурирования (Отдел текстурирования)

Исполнители подпроцесса

* Главный концепт-художник (Отдел концепта)
* Начальник отдела VFX (Отдел VFX)
* Начальник отдела анимации (Отдел анимации)
* Начальник отдела композитинга (Отдел композитинга)
* Начальник отдела моделирования (Отдел моделирования)
* Начальник отдела рендеринга (Отдел рендеринга)
* Начальник отдела сетапа (Отдел сетапа)
* Начальник отдела текстурирования (Отдел текстурирования)

Начало выполнения

Начало выполнения процесса сразу после планирования задач.

Требования к срокам

Требования к срокам не более 10 минут

Результат выполнения

Результатом выполнения процесса является готовые локальные ТЗ.

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Задачи | Список задач | Начальники | А2.2 Планирование задач |
| 2. | ОТЗ | Общее ТЗ | Начальники | А1.5 Установка сроков сдачи |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Локальные ТЗ | ТЗ на концепты | Концепт-художники окружений  Концепт-художники персонажей | А3.1 Создание концепта |
| ТЗ на персонажей | Аниматоры  Сетап-художники  Текстураторы  Художник по волосам  Художник по тканям  Художник твердых тел  Цифровой скульптер | А3.2 Разработка персонажей |
| ТЗ на окружение | Текстураторы  Художник по процедурной генерации  Художник по процедурному моделированию  Художник по растениям  Художник твердых тел | А3.3 Разработка окружений |
| ТЗ на визуальные эффекты | Художник по процедурной генерации  Художник по разрушениям  Художник по толпам  Художник по флюидам | А3.4 Создание визуальных эффектов |
| ТЗ на визуализацию | Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |

| № | Роль | Субъект | Подразделение | Предмет деятельности |
| --- | --- | --- | --- | --- |
| 1. | Начальники | Главный концепт-художник | Отдел концепта |  |
| Начальник отдела VFX | Отдел VFX |  |
| Начальник отдела анимации | Отдел анимации |  |
| Начальник отдела композитинга | Отдел композитинга |  |
| Начальник отдела моделирования | Отдел моделирования |  |
| Начальник отдела рендеринга | Отдел рендеринга |  |
| Начальник отдела сетапа | Отдел сетапа |  |
| Начальник отдела текстурирования | Отдел текстурирования |  |

А3 Реализация проекта

Содержание деятельности

Содержанием деятельности по процессу «А3 Реализация проекта» является выполнение работ, связанных с реализацией проекта. Создаются 3D-модели, текстуры и анимации. Все это собирается вместе и визуализируется.

Исполнители процесса

Исполнителями процесса являются:

* Отдел разработки

Начало выполнения процесса

Начало выполнения процесса определяется в общем ТЗ.

Результат процесса

Основным результатом процесса является графический контент, тематика которого определяется в общем ТЗ.

Требования к срокам выполнения процесса

определяется в общем ТЗ

Документация процесса

Выполнение процесса «Реализация проекта» регламентируется следующей плановой или нормативно-методической документацией:

|  |
| --- |
| * ТЗ на визуализацию |
| * ТЗ на визуальные эффекты |
| * ТЗ на концепты |
| * ТЗ на окружение |
| * ТЗ на персонажей |

ВЗАИМОДЕЙСТВИЕ С ДРУГИМИ ПРОЦЕССАМИ И ВНЕШНЕЙ СРЕДОЙ

При выполнении процесса «А3 Реализация проекта» осуществляется взаимодействие с другими процессами и с внешней средой по входу, выходу, управлению и механизмам:

Взаимодействие по Входу

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Задачи | Список задач | Начальники | А2.2 Планирование задач |

Взаимодействие по Выходу

| № | Выход | Объекты | Передается | |
| --- | --- | --- | --- | --- |
| Получатель | Процесс |
| 1. | Графический контент | Реализованный графический контент |  | {Граница} |
| Менеджер | А4.1 Отправка заказчику |

Управление процесса

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Локальные ТЗ | ТЗ на визуализацию  ТЗ на визуальные эффекты  ТЗ на концепты  ТЗ на окружение  ТЗ на персонажей | Начальники | А2.3 Формирование локальных ТЗ |

Механизмы процесса

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Отдел разработки |  |  | {Граница} |

ОРГАНИЗАЦИЯ ВЫПОЛНЕНИЯ ПРОЦЕССА

А3.1 Создание концепта

Содержанием деятельности по процессу «А3.1 Создание концепта» является отрисовка концепт-артов для персонажей и окружений, создание ключевых кадров и раскадровка каждой сцены проекта.

Владелец подпроцесса

* Главный концепт-художник (Отдел концепта)

Исполнители подпроцесса

* Концепт-художники окружений (Отдел концепта)
* Концепт-художники персонажей (Отдел концепта)

Начало выполнения

Начало выполнения процесса после формирования локальных ТЗ.

Требования к срокам

Требования к срокам определяется локальным ТЗ

Результат выполнения

Результатом выполнения процесса является концепт арты и раскадровки.

Документация подпроцесса

Выполнение подпроцесса регламентируется следующей плановой и нормативно-методической документацией:

|  |
| --- |
| * ТЗ на концепты |

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Задачи | Список задач | Начальники | А2.2 Планирование задач |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Концепты | Концепт-арты | Аниматоры  Сетап-художники  Текстураторы  Художник по волосам  Художник по тканям  Художник твердых тел  Цифровой скульптер | А3.2 Разработка персонажей |
| Текстураторы  Художник по процедурной генерации  Художник по процедурному моделированию  Художник по растениям  Художник твердых тел | А3.3 Разработка окружений |
| Художник по процедурной генерации  Художник по разрушениям  Художник по толпам  Художник по флюидам | А3.4 Создание визуальных эффектов |
| Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |
| 2. | Раскадровки | Раскадровки | Аниматоры  Сетап-художники  Текстураторы  Художник по волосам  Художник по тканям  Художник твердых тел  Цифровой скульптер | А3.2 Разработка персонажей |

Управление подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | ТЗ на концепты | ТЗ на концепты | Начальники | А2.3 Формирование локальных ТЗ |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Отдел концептинга |  |  | {Граница} |

А3.2 Разработка персонажей

Содержанием деятельности по процессу «А3.2 Разработка персонажей» является моделирование, текстурация, оснащений и анимация персонажей.

Владелец подпроцесса

* Начальник отдела анимации (Отдел анимации)
* Начальник отдела моделирования (Отдел моделирования)
* Начальник отдела сетапа (Отдел сетапа)
* Начальник отдела текстурирования (Отдел текстурирования)

Исполнители подпроцесса

* Аниматоры (Отдел анимации)
* Сетап-художники (Отдел сетапа)
* Текстураторы (Отдел текстурирования)
* Художник по волосам (Отдел VFX)
* Художник по тканям (Отдел моделирования)
* Художник твердых тел (Отдел моделирования)
* Цифровой скульптер (Отдел моделирования)

Начало выполнения

Начало выполнения процесса с момента поступления концептов и раскадровок.

Требования к срокам

Требования к срокам определяется локальным ТЗ

Результат выполнения

Результатом выполнения процесса является готовые ассеты персонажей.

Документация подпроцесса

Выполнение подпроцесса регламентируется следующей плановой и нормативно-методической документацией:

|  |
| --- |
| * ТЗ на персонажей |

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Задачи | Список задач | Начальники | А2.2 Планирование задач |
| 2. | Концепты | Концепт-арты | Концепт-художники окружений  Концепт-художники персонажей | А3.1 Создание концепта |
| 3. | Раскадровки | Раскадровки | Концепт-художники окружений  Концепт-художники персонажей | А3.1 Создание концепта |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Асеты персонажей | Асеты персонажей | Художник по процедурной генерации  Художник по разрушениям  Художник по толпам  Художник по флюидам | А3.4 Создание визуальных эффектов |
| Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |

Управление подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | ТЗ на персонажей | ТЗ на персонажей | Начальники | А2.3 Формирование локальных ТЗ |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Отдел анимации |  |  | {Граница} |
| 2. | Отдел моделирования |  |  | {Граница} |
| 3. | Отдел сетапа |  |  | {Граница} |
| 4. | Отдел текстурирования |  |  | {Граница} |

А3.3 Разработка окружений

Содержанием деятельности по процессу «А3.3 Разработка окружений» является моделирование и текстурация объектов окружения.

Владелец подпроцесса

* Начальник отдела VFX (Отдел VFX)
* Начальник отдела моделирования (Отдел моделирования)
* Начальник отдела текстурирования (Отдел текстурирования)

Исполнители подпроцесса

* Текстураторы (Отдел текстурирования)
* Художник по процедурной генерации (Отдел VFX)
* Художник по процедурному моделированию (Отдел моделирования)
* Художник по растениям (Отдел моделирования)
* Художник твердых тел (Отдел моделирования)

Начало выполнения

Начало выполнения процесса с момента поступления концептов и раскадровок.

Требования к срокам

Требования к срокам определяется локальным ТЗ

Результат выполнения

Результатом выполнения процесса является готовые асеты окружений.

Документация подпроцесса

Выполнение подпроцесса регламентируется следующей плановой и нормативно-методической документацией:

|  |
| --- |
| * ТЗ на окружение |

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Задачи | Список задач | Начальники | А2.2 Планирование задач |
| 2. | Концепты | Концепт-арты | Концепт-художники окружений  Концепт-художники персонажей | А3.1 Создание концепта |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Асеты окружений | Асеты окружений | Художник по процедурной генерации  Художник по разрушениям  Художник по толпам  Художник по флюидам | А3.4 Создание визуальных эффектов |
| Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |

Управление подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | ТЗ на окружение | ТЗ на окружение | Начальники | А2.3 Формирование локальных ТЗ |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Отдел VFX |  |  | {Граница} |
| 2. | Отдел моделирования |  |  | {Граница} |
| 3. | Отдел текстурирования |  |  | {Граница} |

А3.4 Создание визуальных эффектов

Содержанием деятельности по процессу «А3.4 Создание визуальных эффектов» является создание симуляций, взрывов, водных эффектов, разрушений и пр.

Владелец подпроцесса

* Начальник отдела VFX (Отдел VFX)

Исполнители подпроцесса

* Художник по процедурной генерации (Отдел VFX)
* Художник по разрушениям (Отдел VFX)
* Художник по толпам (Отдел VFX)
* Художник по флюидам (Отдел VFX)

Начало выполнения

Начало выполнения процесса с момента поступления ассетов окружений и персонажей.

Требования к срокам

Требования к срокам определяется локальным ТЗ

Результат выполнения

Результатом выполнения процесса является асеты эффектов.

Документация подпроцесса

Выполнение подпроцесса регламентируется следующей плановой и нормативно-методической документацией:

|  |
| --- |
| * ТЗ на визуальные эффекты |

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Асеты окружений | Асеты окружений | Текстураторы  Художник по процедурной генерации  Художник по процедурному моделированию  Художник по растениям  Художник твердых тел | А3.3 Разработка окружений |
| 2. | Асеты персонажей | Асеты персонажей | Аниматоры  Сетап-художники  Текстураторы  Художник по волосам  Художник по тканям  Художник твердых тел  Цифровой скульптер | А3.2 Разработка персонажей |
| 3. | Задачи | Список задач | Начальники | А2.2 Планирование задач |
| 4. | Концепты | Концепт-арты | Концепт-художники окружений  Концепт-художники персонажей | А3.1 Создание концепта |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Эффекты | Кэш визуальных эффектов | Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |

Управление подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | ТЗ на эффекты | ТЗ на визуальные эффекты | Начальники | А2.3 Формирование локальных ТЗ |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Отдел VFX |  |  | {Граница} |

А3.5 Визуализация и сборка

Содержанием деятельности по процессу «А3.5 Визуализация и сборка» является визуализация сцен проекта, композитинг компонентов и формирование выходного материала.

Владелец подпроцесса

* Начальник отдела композитинга (Отдел композитинга)

* Начальник отдела рендеринга (Отдел рендеринга)

Исполнители подпроцесса

* Монтажеры (Отдел композитинга)
* Специалист по интерактивным решениям (Отдел рендеринга)
* Специалист по рендерингу (Отдел рендеринга)
* Художник по освещению (Отдел рендеринга)
* Художники по композитингу (Отдел композитинга)
* Художники по цветокоррекции (Отдел композитинга)
* Шейдер-программист (Отдел рендеринга)

Начало выполнения

Начало выполнения процесса с момента поступления ассетов окружений и персонажей.

Требования к срокам

Требования к срокам определяется локальным ТЗ

Результат выполнения

Результатом выполнения процесса является реализованный проект.

Документация подпроцесса

Выполнение подпроцесса регламентируется следующей плановой и нормативно-методической документацией:

|  |
| --- |
| * ТЗ на визуализацию |

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Асеты окружений | Асеты окружений | Текстураторы  Художник по процедурной генерации  Художник по процедурному моделированию  Художник по растениям  Художник твердых тел | А3.3 Разработка окружений |
| 2. | Асеты персонажей | Асеты персонажей | Аниматоры  Сетап-художники  Текстураторы  Художник по волосам  Художник по тканям  Художник твердых тел  Цифровой скульптер | А3.2 Разработка персонажей |
| 3. | Задачи | Список задач | Начальники | А2.2 Планирование задач |
| 4. | Концепты | Концепт-арты | Концепт-художники окружений  Концепт-художники персонажей | А3.1 Создание концепта |
| 5. | Эффекты | Кэш визуальных эффектов | Художник по процедурной генерации  Художник по разрушениям  Художник по толпам  Художник по флюидам | А3.4 Создание визуальных эффектов |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Графический контент | Реализованный графический контент |  | {Граница} |
| Менеджер | А4.1 Отправка заказчику |

Управление подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | ТЗ на визуализацию | ТЗ на визуализацию | Начальники | А2.3 Формирование локальных ТЗ |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Отдел композитинга |  |  | {Граница} |
| 2. | Отдел рендеринга |  |  | {Граница} |

СОСТАВ НАБОРОВ ОБЪЕКТОВ И РОЛЕЙ, УЧАСТВУЮЩИХ В ВЫПОЛНЕНИИ ПРОЦЕССА

| № | Набор объектов | Объекты, входящие в состав набора |
| --- | --- | --- |
| 1. | Асеты окружений |  |
| 2. | Асеты персонажей |  |
| 3. | Концепт-арты |  |
| 4. | Раскадровки |  |
| 5. | Реализованный графический контент |  |

| № | Роль | Субъект | Подразделение | Предмет деятельности |
| --- | --- | --- | --- | --- |
| 1. | Начальники | Главный концепт-художник | Отдел концепта |  |
| Начальник отдела VFX | Отдел VFX |  |
| Начальник отдела анимации | Отдел анимации |  |
| Начальник отдела композитинга | Отдел композитинга |  |
| Начальник отдела моделирования | Отдел моделирования |  |
| Начальник отдела рендеринга | Отдел рендеринга |  |
| Начальник отдела сетапа | Отдел сетапа |  |
| Начальник отдела текстурирования | Отдел текстурирования |  |

А4 Отправка заказчику и доработка

Содержание деятельности

Содержанием деятельности по процессу «А4 Отправка заказчику и доработка» является отправка заказчику реализованного проекта и выполнение доработки на основе списка правок.

Исполнители процесса

Исполнителями процесса являются:

* Менеджер
* Отдел разработки

Начало выполнения процесса

Начало выполнения процесса после реализации проекта.

Результат процесса

Основным результатом процесса является финальный продукт.

Требования к срокам выполнения процесса

не более 1 дня

ВЗАИМОДЕЙСТВИЕ С ДРУГИМИ ПРОЦЕССАМИ И ВНЕШНЕЙ СРЕДОЙ

При выполнении процесса «А4 Отправка заказчику и доработка» осуществляется взаимодействие с другими процессами и с внешней средой по входу, выходу, управлению и механизмам:

Взаимодействие по Входу

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Графический контент | Реализованный графический контент | Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |

Механизмы процесса

| № | Вход | Объекты | Поступает от | |
| --- | --- | --- | --- | --- |
| Исполнитель | Процесс |
| 1. | Менеджер |  |  | {Граница} |
| 2. | Отдел разработки |  |  | {Туннель} |

ОРГАНИЗАЦИЯ ВЫПОЛНЕНИЯ ПРОЦЕССА

А4.1 Отправка заказчику

Содержанием деятельности по процессу «А4.1 Отправка заказчику» является отправка ссылки на видео или же готовый материал заказчику.

Владелец подпроцесса

* Менеджер

Исполнители подпроцесса

* Менеджер

Начало выполнения

Начало выполнения процесса сразу после формирования материала.

Требования к срокам

Требования к срокам не более часа

Результат выполнения

Результатом выполнения процесса является список правок.

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Графический контент | Реализованный графический контент | Монтажеры  Специалист по интерактивным решениям  Специалист по рендерингу  Художник по освещению  Художники по композитингу  Художники по цветокоррекции  Шейдер-программист | А3.5 Визуализация и сборка |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Правки | Список правок |  | А4.2 Поиск дорабатываемых ассетов |
| Отдел разработки | А4.3 Изменение ассетов |
| Отдел композитинга  Отдел рендеринга | А4.4 Визуализация и сборка |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Менеджер |  |  | {Граница} |

А4.2 Поиск дорабатываемых ассетов

Содержанием деятельности по процессу «А4.2 Поиск дорабатываемых ассетов» является поиск асетов, требующих доработки или корректировки.

Владелец подпроцесса

* Менеджер

Начало выполнения

Начало выполнения процесса сразу после получения правок.

Требования к срокам

Требования к срокам не более 10 минут

Результат выполнения

Результатом выполнения процесса является список ассетов.

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Правки | Список правок | Менеджер | А4.1 Отправка заказчику |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Список ассетов |  | Отдел разработки | А4.3 Изменение ассетов |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Начальники отделов |  |  | {Туннель} |

А4.3 Изменение ассетов

Содержанием деятельности по процессу «А4.3 Изменение ассетов» является внесение изменений в список найденных асетов.

Исполнители подпроцесса

* Отдел разработки

Документация подпроцесса

Выполнение подпроцесса регламентируется следующей плановой и нормативно-методической документацией:

|  |
| --- |
| * Список правок |

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Список ассетов |  |  | А4.2 Поиск дорабатываемых ассетов |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Доработанные асеты |  | Отдел композитинга  Отдел рендеринга | А4.4 Визуализация и сборка |

Управление подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Правки | Список правок | Менеджер | А4.1 Отправка заказчику |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Отдел окружений |  |  | {Туннель} |
| 2. | Отдел персонажей |  |  | {Туннель} |

А4.4 Визуализация и сборка

Содержанием деятельности по процессу «А4.4 Визуализация и сборка» является визуализация сцен проекта, композитинг компонентов и формирование выходного материала.

Исполнители подпроцесса

* Отдел композитинга (Отдел разработки)
* Отдел рендеринга (Отдел разработки)

Начало выполнения

Начало выполнения процесса сразу после изменения асетов.

Требования к срокам

Требования к срокам определяется локальным ТЗ

Результат выполнения

Результатом выполнения процесса является готовый графический контент.

Документация подпроцесса

Выполнение подпроцесса регламентируется следующей плановой и нормативно-методической документацией:

|  |
| --- |
| * Список правок |

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Доработанные асеты |  | Отдел разработки | А4.3 Изменение ассетов |

Выходы подпроцесса

| № | Выход | Объекты | Получатель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Исправленный контент |  | Менеджер | А4.5 Отправка готового проекта |

Управление подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Правки | Список правок | Менеджер | А4.1 Отправка заказчику |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Отдел сборки |  |  | {Туннель} |

А4.5 Отправка готового проекта

Содержанием деятельности по процессу «А4.5 Отправка готового проекта» является отправка ссылки на видео или же готовый материал заказчику.

Исполнители подпроцесса

* Менеджер

Входы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Исправленный контент |  | Отдел композитинга  Отдел рендеринга | А4.4 Визуализация и сборка |

Механизмы подпроцесса

| № | Вход | Объекты | Исполнитель | Процесс |
| --- | --- | --- | --- | --- |
| 1. | Менеджер |  |  | {Граница} |

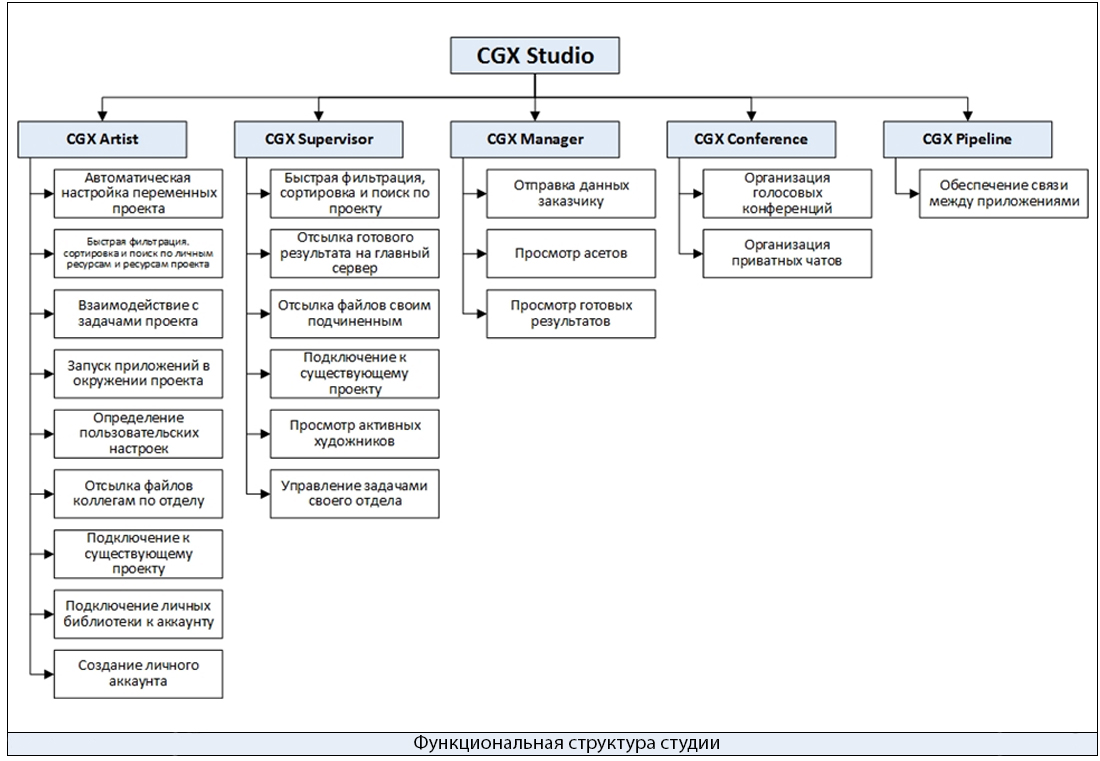
СОСТАВ НАБОРОВ ОБЪЕКТОВ И РОЛЕЙ, УЧАСТВУЮЩИХ В ВЫПОЛНЕНИИ ПРОЦЕССА

| № | Набор объектов | Объекты, входящие в состав набора |
| --- | --- | --- |
| 1. | Реализованный графический контент |  |

1. Описание автоматизируемых функций

Описание автоматизируемых функций системы

«CGX Studio»



«CGX Supervisor»

Данная подсистема предназначена начальникам отделов. Она взаимодействует с подсистемой «CGX Artist» через модуль «CGX Pipeline» и реализует следующие функции:

* быстрая фильтрация, сортировка и поиск по проекту;
* отсылка готового результата на главный сервер;
* отсылка файлов своим подчиненным;
* подключение к существующему проекту;
* просмотр активных художников;
* управление задачами своего отдела.

Быстрая фильтрация, сортировка и поиск по проекту

Данная функция обеспечивает быструю фильтрацию, сортировку и поиск файлов проекта. Поиск осуществляется по шаблону регулярного выражения, фильтрация – объектная (фильтр моделей, фильтр текстур и т.д.). Сортировка – стандартная (порядок возрастания и убывания) и объектная (сначала модели, текстуры и т.д.)

Временной регламент: в период подключения к проекту

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.5 Визуализация и сборка; | |
| А4.2 Поиск дорабатываемых ассетов | |
| А4.3 Изменение ассетов | |
| Входные данные | Выходные данные |
| Шаблон регулярного выражения, параметры сортировки и фильтрации | Отфильтрованные и отсортированные файлы/асеты проекта |
| Формы представления | |
| Видеокадр «АРМ Начальника. Асетная структура проекта» - рис. 2.1 в [1] | |
| Видеокадр «АРМ Начальника. Файловая структура проекта» - рис. 2.2 в [1] | |

Отсылка готового результата на главный сервер

Данная функция позволяет начальнику отдела оперативно отправлять асеты/файлы на главный сервер. Для этого требуется лишь выделить нужное и нажать на кнопку «Отправить выбранные файлы». Приложение «CGX Supervisor» уведомит приложение «CGX Pipeline» (на главном сервере), после чего «CGX Pipeline» создаст необходимый раздел в файловой структуре проекта и примет отправленные файлы.

Временной регламент: в период подключения к проекту на главном сервере.

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.1 Создание концепта | |
| А3.2 Разработка персонажей | |
| А3.3 Разработка окружений | |
| А3.4 Создание визуальных эффектов | |
| А3.5 Визуализация и сборка | |
| Входные данные | Выходные данные |
| Асет/файл любого формата | Асет/файл, отправленный на главный сервер |
| Формы представления | |
| Видеокадр «АРМ Начальника. Асетная структура проекта» - рис. 2.1 в [1] | |
| Видеокадр «АРМ Начальника. Файловая структура проекта» - рис. 2.2 в [1] | |

Отсылка файлов своим подчиненным

Данная функция позволяет начальнику отдела оперативно отправлять асеты/файлы на своим подчиненным. Для этого требуется выбрать нужных художников и списка, выделить нужные файлы/асеты и нажать на кнопку «Отправить». В приложении «CGX Artist» появится уведомление, что были получены файлы от начальника.

Временной регламент: в период подключения художников к проекту на сервере отдела.

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.1 Создание концепта | |
| А3.2 Разработка персонажей | |
| А3.3 Разработка окружений | |
| А3.4 Создание визуальных эффектов | |
| А3.5 Визуализация и сборка | |
| Входные данные | Выходные данные |
| Асет/файл любого формата | Асет/файл, отправленный на главный сервер |
| Формы представления | |
| Видеокадр «АРМ Начальника. Отправка файлов подчиненным» - рис. 2.5 в [1] | |

Подключение к существующему проекту;

Данная функция позволяет начальнику отдела оперативно подключаться к проектам на главном сервере и создавать их инстансы на сервере своего отдела. Для создания инстанса необходимо:

* подключиться к проекту на главном сервере;
* нажать на кнопку «Создать на сервере отдела»

Временной регламент: в период подключения к главному серверу.

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.1 Создание концепта | |
| А3.2 Разработка персонажей | |
| А3.3 Разработка окружений | |
| А3.4 Создание визуальных эффектов | |
| А3.5 Визуализация и сборка | |
| Входные данные | Выходные данные |
| Проект, созданный менеджером на главном сервере | Локальный проект (инстанс), созданный на сервере отдела |
| Формы представления | |
| Видеокадр «АРМ Начальника. Файловая структура проекта» - рис. 2.2 в [1] | |

Просмотр активных художников

Данная функция позволяет начальнику отдела просматривать активных художников в конкретных проектах. Для получения списка художников необходимо выбрать проект.

Временной регламент: в период подключения к проекту на главном сервере.

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.1 Создание концепта | |
| А3.2 Разработка персонажей | |
| А3.3 Разработка окружений | |
| А3.4 Создание визуальных эффектов | |
| А3.5 Визуализация и сборка | |
| Входные данные | Выходные данные |
| Локальный проект, созданный на сервере отдела | Список художников, работающих над проектом |
| Формы представления | |
| Видеокадр «АРМ Начальника. Активные художники» - рис. 2.4 в [1] | |

Управление задачами своего отдела

Данная функция даёт возможность начальникам отделов создавать, редактировать и удалять задачи для своего отдела. После каждой операции над задачей оповещается, выполняющий её, художник (автоматически).

Временной регламент: в период подключения к проекту на главном сервере.

|  |  |
| --- | --- |
| Технологические процессы | |
| А2.1 Постановка задач | |
| А2.2 Планирование задач | |
| Входные данные | Выходные данные |
| Проект, созданный менеджером | Список сформированных задач |
| Проект, созданный на сервере отдела |
| Формы представления | |
| Видеокадр «АРМ Начальника. Задачи» - рис. 2.3 в [1] | |

«CGX Artist»

Данная подсистема предназначена художникам отделов. Она взаимодействует с подсистемой «CGX Supervisor» через модуль «CGX Pipeline» и реализует следующие функции:

* Создание личного аккаунта;
* Автоматическая настройка переменных проекта;
* Быстрая фильтрация, сортировка и поиск по личным ресурсам и ресурсам проекта;
* Взаимодействие с задачами проекта;
* Запуск приложений в окружении проекта;
* Быстрый просмотр hdr и exr файлов;
* Определение пользовательских настроек;
* Отсылка файлов коллегам по отделу;
* Подключение к существующему проекту;
* Подключение личной библиотеки к аккаунту.

Создание личного аккаунта

Данная функция обеспечивает регистрацию художника в системе.

Временной регламент: в любое время при наличии связи с сервером отдела

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Логин, пароль, ФИО и должность | Список художников, работающих над проектом |
| Формы представления | |
| Видеокадр «АРМ Художника. Вход в систему» - рис. 1.1 в [1] | |
| Видеокадр «АРМ Художника. Регистрация в системе» - рис. 1.2 в [1] | |

Автоматическая настройка переменных проекта

Данная функция определяет (либо переопределяет) необходимые переменные окружения в операционной системе в зависимости от корневой директории проекта, созданного на компьютере художника.

Временной регламент: сразу после подключения к проекту.

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.1 Создание концепта | |
| А3.2 Разработка персонажей | |
| А3.3 Разработка окружений | |
| А3.4 Создание визуальных эффектов | |
| А3.5 Визуализация и сборка | |
| Входные данные | Выходные данные |
| Локальный проект, созданный на сервере отдела | Список переменных окружения, определенных в операционной системе |
| Формы представления | |
| Данная функция является автоматической, поэтому не имеет форм представления | |

Быстрая фильтрация, сортировка и поиск по личным ресурсам и ресурсам проекта

Данная функция обеспечивает быструю фильтрацию, сортировку и поиск файлов проекта и личной библиотеки. Поиск осуществляется по шаблону регулярного выражения, фильтрация – объектная (фильтр моделей, фильтр текстур и т.д.). Сортировка – стандартная (порядок возрастания и убывания) и объектная (сначала модели, текстуры и т.д.)

Временной регламент: в период подключения к проекту

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.5 Визуализация и сборка; | |
| А4.2 Поиск дорабатываемых ассетов | |
| А4.3 Изменение ассетов | |
| Входные данные | Выходные данные |
| Шаблон регулярного выражения, параметры сортировки и фильтрации | Отфильтрованные и отсортированные файлы/асеты проекта |
| Формы представления | |
| Видеокадр «АРМ Художника. Асетная структура проекта» - рис. 1.5 в [1] | |
| Видеокадр «АРМ Художника. Файловая структура проекта». - рис. 1.6 в [1] | |

Взаимодействие с задачами проекта

Данная функция позволяет художникам брать задачи, сформированные начальником, и отправлять результаты на проверку. Задачи отображаются в виде списка. Если задача была удалена, добавлена или модифицирована, художник получает уведомление с комментарием. После выполнения задачи, следует отправить результат начальнику отдела и дождаться комментирования проделанной работы. Все оповещения будут находиться в поле оповещений в нижней части окна программы.

Временной регламент: после появления задач на сервере отдела.

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.1 Создание концепта | |
| А3.2 Разработка персонажей | |
| А3.3 Разработка окружений | |
| А3.4 Создание визуальных эффектов | |
| А3.5 Визуализация и сборка | |
| Входные данные | Выходные данные |
| Задачи, сформированные начальником | Список взятых задач, комментарии к выполненным задачам |
| Формы представления | |
| Видеокадр «АРМ Художника. Задачи» - рис. 1.8 в [1] | |

Запуск приложений в окружении проекта

Данная функция обеспечивает автоматическую настройку переменных окружения программного пакета на этапе запуска. Это позволяет определить различные параметры приложения (доступные и недоступные внутри приложения) для удобной работы над конкретным проектом (например, определение директорий хранения временного кэша симуляций или анимаций).

Временной регламент: после подключения к проекту.

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.1 Создание концепта | |
| А3.2 Разработка персонажей | |
| А3.3 Разработка окружений | |
| А3.4 Создание визуальных эффектов | |
| А3.5 Визуализация и сборка | |
| Входные данные | Выходные данные |
| Приложение, требуемое для запуска, проект. | Запущенное приложение настроенное под текущий проект |
| Формы представления | |
| Видеокадр «АРМ Художника. Структура рабочих файлов» - рис. 1.7 в [1] | |
| Видеокадр «АРМ Художника. Подключенные программы» - рис. 1.9 в [1] | |

Быстрый просмотр hdr и exr файлов

Данная функция позволяет художникам незамедлительно просматривать тяжёлые hdr и exr изображения. После добавления контента (или подключения личной библиотеки) «CGX Artist» автоматически создает, сконвертированные в .jpeg формат, копии всех hdr и exr изображений. Для просмотра используется копия, а при копировании в буфер обмена попадает ссылка на исходное изображение.

Временной регламент: при наличии hdr и exr изображений в проекте или библиотеки.

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.2 Разработка персонажей | |
| А3.3 Разработка окружений | |
| А3.4 Создание визуальных эффектов | |
| А3.5 Визуализация и сборка | |
| Входные данные | Выходные данные |
| hdr или exr изображение | Быстрый просмотр изображения |
| Формы представления | |
| Видеокадр «АРМ Художника. Файловая структура проекта» - рис. 1.6 в [1] | |
| Видеокадр «АРМ Художника. Файловая структура личной библиотеки» - рис. 3 в [1] | |

Определение пользовательских настроек

Данная функция позволяет подключить программы и личную библиотеку, а также создать пользовательские переменные.

Временной регламент: весь период в авторизованной системе.

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Параметры настроек | Настроенный аккаунт художника |
| Формы представления | |
| Видеокадр «АРМ Художника. Настройки» - рис. 1.12 в [1] | |

Отсылка файлов коллегам по отделу

Данная задача позволяет художникам передавать файлы и асеты (размером до 5 ГБ) своим коллегам. Для передачи следует выделить необходимые файлы/асеты, выбрать необходимых коллег из списка и нажать на кнопку «Отправить выбранные файлы».

Временной регламент: при подключении к серверу отдела.

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.2 Разработка персонажей | |
| А3.3 Разработка окружений | |
| А3.4 Создание визуальных эффектов | |
| А3.5 Визуализация и сборка | |
| Входные данные | Выходные данные |
| Файл/асет, требуемый для передачи | Переданный файл/асет |
| Формы представления | |
| Видеокадр «АРМ Художника. Асетная структура проекта» - рис. 1.5 в [1] | |
| Видеокадр «АРМ Художника. Файловая структура проекта» - рис. 1.6 в [1] | |
| Видеокадр «АРМ Художника. Структура рабочих файлов» - рис. 1.7 в [1] | |
| Видеокадр «АРМ Художника. Асеты личной библиотеки» - рис. 1.10 в [1] | |
| Видеокадр «АРМ Художника. Файловая структура личной библиотеки» - рис. 1.11 в [1] | |

Подключение к существующему проекту

Данная функция обеспечивает быстрое подключение к проекту на локальном сервере отдела. После подключения происходит автоматическая перенастройка (в пакетном режиме) необходимых переменных в операционной системе.

Временной регламент: после создания проекта на сервере отдела.

|  |  |
| --- | --- |
| Технологические процессы | |
| А3.1 Создание концепта | |
| А3.2 Разработка персонажей | |
| А3.3 Разработка окружений | |
| А3.4 Создание визуальных эффектов | |
| А3.5 Визуализация и сборка | |
| Входные данные | Выходные данные |
| Проект на сервере отдела | Файловая структура подключенного проекта |
| Формы представления | |
| Видеокадр «АРМ Художника. Файловая структура проекта» - рис. 1.6 в [1] | |

Подключение личной библиотеки к аккаунту

Данная задача позволяет художникам подключать личную библиотеку контента для быстрого доступа к ней, а также для ее управления. Подключение осуществляется в разделе настроек.

Временной регламент: весь период после авторизации.

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Ссылка на библиотеку | Подключенная библиотека к аккаунту |

«CGX Conference»

Данное приложение способно организовывать голосовые конференции с настройкой канала связи и работать по принципу push-to-talk (нажми, чтобы сказать), создавать публичные и приватные чаты для обмена текстовыми сообщениями.

Функции приложения:

* Организация голосовых конференций
* Приватные чаты

Организация голосовых конференций

Данная функция позволяет начальникам и художникам находиться в постоянном общении, что позволяет быстро решать многие вопросы.

Временной регламент: в любой момент рабочей смены.

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Художники/начальники | Голосовая конференция |
| Формы представления | |
| Видеокадр «Discord. Голосовая конференция» - рис. 4.1 в [1] | |

Организация приватных чатов

Данная функция позволяет художникам решать вопросы «с глазу на глаз», чтобы не отвлекать занятых коллег или начальника.

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Художники/начальники | Голосовой звонок |

«CGX Manager»

Данная подсистема предназначается менеджеру отдела разработки, она дает ему возможность создать проект, над которым будет работать его отдел, просматривать асеты проекта и финальные результаты. Так же эта подсистема позволит ему отправлять скриншоты асетов и готовые результаты заказчику. Есть 2 варианта просмотра асетов и результатов:

* по проектам;
* по конкретному заказчику.

Функции подсистемы:

* отправка данных заказчику;
* просмотр ассетов;
* просмотр готовых результатов.

Отправка данных заказчику

Данная функция позволяет менеджеру отправлять скриншоты ассетов и готовые результаты заказчику.

Временной регламент: после создания асетов/после реализации проекта.

|  |  |
| --- | --- |
| Технологические процессы | |
| А4 Отправка заказчику и доработка | |
| Входные данные | Выходные данные |
| Готовые асеты/реализованный проект | Отправленные данные заказчику |
| Формы представления | |
| Видеокадр «Менеджер проектов. Просмотр асетов по проектам» - рис. 3.1 в [1] | |
| Видеокадр «Менеджер проектов. Просмотр результатов по проектам» - рис. 3.2 в [1] | |
| Видеокадр «Менеджер проектов. Просмотр асетов конкретного заказчика» - рис. 3.3 в [1] | |
| Видеокадр «Менеджер проектов. Просмотр результатов проекта конкретного заказчика» - рис. 3.4 в [1] | |

Просмотр ассетов и готовых результатов

Данная задача позволяет менеджеру просматривать готовые асеты и готовые результаты по проектам.

Временной регламент: после создания проекта на главном сервере.

|  |  |
| --- | --- |
| Технологические процессы | |
| А4 Отправка заказчику и доработка | |
| Входные данные | Выходные данные |
| Готовые асеты/реализованный проект | Отправленные данные заказчику |
| Формы представления | |
| Видеокадр «Менеджер проектов. Просмотр асетов по проектам» - рис. 3.1 в [1] | |
| Видеокадр «Менеджер проектов. Просмотр результатов по проектам» - рис. 3.2 в [1] | |
| Видеокадр «Менеджер проектов. Просмотр асетов конкретного заказчика» - рис. 3.3 в [1] | |
| Видеокадр «Менеджер проектов. Просмотр результатов проекта конкретного заказчика» - рис. 3.4 в [1] | |

«CGX Pipeline»

Данный модель обеспечивает связь между серверами, начальниками и художниками, а также реализует обмен данными между отделами и художниками.

Временной регламент: при подключении начальников и отделов к локальным серверам, а также при подключении локальных серверов к главному серверу (для передачи между локальными серверами).

|  |  |
| --- | --- |
| Технологические процессы | |
| А3 Реализация проекта | |
| А4 Отправка заказчику и доработка | |
| Входные данные | Выходные данные |
| Асеты и файлы, необходимые для передачи | Переданные данные |

1. Спецификации серверов

**Спецификации серверов**

**для системы «CGX Studio»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Спецификация серверов отделов** | | | | |
| **Позиция** | **Наименование и техническая характеристика оборудования** | **Тип, марка оборудования. Обозначение документа и номер опросного листа** | **Ед.**  **измерения** | **Количество** |
| **1** | **2** | **3** | **4** | **5** |
| 2 | ЦПУ | Intel Xeon, 8 ядер | Шт. | 2 |
| 3 | Видеоадаптер, 6 ГБ |  | Шт. | 1 |
| 4 | Оперативная память DDR4, 16 ГБ |  | Шт. | 8 |
| 5 | Блок питания, 700 Вт |  | Шт. | 1 |
| 6 | Сетевая двухпортовая плата, 10 Гбит/с |  | Шт. | 2 |
| 7 | Сетевой адаптер, 10 Гбит/с | GigabitEthernet NC380T PCI Express | Шт. | 1 |
| 8 | Клавиатура рус. |  | Шт. | 1 |
| 9 | Манипулятор типа «мышь» |  | Шт. | 1 |
| 10 | Накопитель на ЖМД, 4 ТБ | WD | Шт. | 2 |
| 6 | Монитор, SVGA, 15`` |  | Шт. | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Спецификация главного сервера** | | | | |
| **Позиция** | **Наименование и техническая характеристика оборудования** | **Тип, марка оборудования. Обозначение документа и номер опросного листа** | **Ед.**  **измерения** | **Количество** |
| **1** | **2** | **3** | **4** | **5** |
| 2 | ЦПУ | Intel Xeon, 16 ядер | Шт. | 2 |
| 3 | Видеоадаптер, 6 ГБ |  | Шт. | 1 |
| 4 | Оперативная память DDR4, 16 ГБ |  | Шт. | 16 |
| 5 | Блок питания, 700 Вт |  | Шт. | 1 |
| 6 | Сетевая двухпортовая плата, 10 Гбит/с |  | Шт. | 2 |
| 7 | Сетевой адаптер, 10 Гбит/с | GigabitEthernet NC380T PCI Express | Шт. | 2 |
| 8 | Клавиатура рус. |  | Шт. | 2 |
| 9 | Манипулятор типа «мышь» |  | Шт. | 1 |
| 10 | Накопитель на ЖМД, 4 ТБ | WD | Шт. | 4 |
| 6 | Монитор, SVGA, 15`` |  | Шт. | 1 |

1. Листинг некоторых классов системы

**Листинг некоторых классов системы «CGX Studio»**

**API.Core.Types.Artist.py**

class Artist(object):

LOGIN = 'login'

PASSWORD = 'password'

NAME = 'name'

SURNAME = 'surname'

def \_\_init\_\_(self, login='', password='', name='', surname=''):

"""

This class is data structure for

users of application.

"""

super(Artist, self).\_\_init\_\_()

self.\_login = login

self.\_password = password

self.\_name = name

self.\_surname = surname

def \_\_eq\_\_(self, other):

try:

return self.\_login == other.login

except:

return False

def \_\_str\_\_(self):

return 'Artist({0}, {1})'.format(self.\_login, self.\_password)

@property

def login(self):

return self.\_login

@property

def password(self):

return self.\_password

@property

def name(self):

return self.\_name

@property

def surname(self):

return self.\_surname

@property

def toDict(self):

return {

self.LOGIN: self.\_login,

self.PASSWORD: self.\_password,

self.NAME: self.\_name,

self.SURNAME: self.\_surname

}

@login.setter

def login(self, login):

if not isinstance(login, str):

raise TypeError("Expected type is < str >")

if login != '': self.\_login = login

@password.setter

def password(self, password):

if not isinstance(password, str):

raise TypeError("Expected type is < str >")

if password != '':

self.\_password = password

@name.setter

def name(self, name):

if not isinstance(name, str):

raise TypeError("Expected type is < str >")

if name != '':

self.\_name = name

@surname.setter

def surname(self, surname):

if not isinstance(surname, str):

raise TypeError("Expected type is < str >")

if surname != '':

self.\_name = surname

def setDict(self, d):

"""

:param d: < dict >

{

'login',

'password',

'name',

'surname'

}

"""

self.login = d[self.LOGIN]

self.password = d[self.PASSWORD]

self.name = d[self.NAME]

self.surname = d[self.SURNAME]

def isIdentical(self, other):

if not isinstance(other, Artist):

raise TypeError("Expected type is 'Artist'")

login = self.\_login == other.login

pasw = self.\_password == other.password

name = self.\_name == other.name

surname = self.\_surname == other.surname

return name and surname and login and pasw

def clear(self):

self.\_name = ''

self.\_surname = ''

self.\_login = ''

self.\_password = ''

**API.Core.Types.Message.py**

class Message(object):

# message types

INFO = 0

QUESTION = 1

SUCCESS = 2

WARNING = 3

ERROR = 4

def \_\_init\_\_(

self,

title='',

type=0,

theme='',

descript='',

):

super(Message, self).\_\_init\_\_()

self.\_\_title = title

self.\_\_type = type

self.\_\_theme = theme

self.\_\_desc = descript

@property

def title(self):

return self.\_\_title

@property

def theme(self):

return self.\_\_theme

@property

def type(self):

return self.\_\_type

@property

def description(self):

return self.\_\_desc

def setTitle(self, text):

self.\_\_title = text

def setType(self, type):

self.\_\_type = type

def setTheme(self, theme):

self.\_\_theme = theme

def setDescription(self, html):

self.\_\_desc = html

def clear(self):

self.\_\_theme = ''

self.\_\_type = 0

self.\_\_title = ''

self.\_\_desc = ''

**API.Core.Types.Program.py**

from Application.API.Core.Containers.VariablesContainer import VariablesContainer

from Application.API.Utility.OS import OSUTIL

class Program(object):

NAME = 'name'

EXEC = 'exe'

EXTENSIONS = 'extensions'

VARIABLES = 'variables'

ARGUMENTS = 'args'

def \_\_init\_\_(self, name='', exe='', exts=[], args=[]):

super(Program, self).\_\_init\_\_()

self.\_name = name

self.\_exe = exe

self.\_extensions = exts

self.\_arguments = args

self.\_variables = VariablesContainer()

def \_\_eq\_\_(self, prog):

try:

name = self.name == prog.login

exe = self.exe == prog.exe

return name and exe

except:

return False

def \_\_str\_\_(self):

string = "{{\n\t{0} | {1}\n\t{3} | {2}\n".format(self.\_name, self.\_exe, self.\_arguments, self.\_extensions)

#

for var in self.variables:

string += "\t" + var.\_\_str\_\_() + "\n"

#

string += "\n}"

return string

def \_\_repr\_\_(self):

return 'Program( {0}, {1} )'.format( self.name, self.exe )

@property

def name(self):

return self.\_name

@property

def exe(self):

return self.\_exe

@property

def extensions(self):

return self.\_extensions

@property

def arguments(self):

return self.\_arguments

@property

def variables(self):

return self.\_variables

@property

def toDict(self):

return {

self.NAME: self.name,

self.EXEC: self.exe,

self.EXTENSIONS: self.extensions,

self.ARGUMENTS: self.arguments

}

@variables.setter

def variables(self, variables: VariablesContainer):

if not isinstance(variables, VariablesContainer):

raise TypeError("Expected type: < VariablesContainer >")

#

self.\_variables = variables

def isExeExist(self):

return OSUTIL.isExist(self.exe)

def setProgram(self, program):

self.\_name = program.name

self.\_exe = program.exe

self.\_extensions = program.extensions

self.\_arguments = program.arguments

self.variables = program.variables

def setDict(self, d):

"""

:param d: <dict{

'install',

'exe',

'icon',

'extensions'

}

>

"""

self.setName(d[self.NAME])

self.setExePath(d[self.EXEC])

self.setExtensions(d[self.EXTENSIONS])

self.setArguments(d[self.ARGUMENTS])

def setArguments(self, args=[]):

self.\_arguments = args

def setName(self, name):

self.\_name = str(name)

def setExePath(self, path):

self.\_exe = str(path)

def setExtensions(self, str\_list: list):

self.\_extensions = str\_list

def clear(self):

self.\_name = ''

self.\_exe = ''

self.\_extensions = []

self.\_arguments = []

self.variables.clear()

**API.Core.Types.Project.py**

from Application.API.Core.Containers.VariablesContainer import VariablesContainer

from Application.API.Core.Types.Project.Metainfo import Metainfo

from Application.API.Core.Types.Variable import Variable

class Project(object):

METAINFO = 'metainfo'

VARIABLES = 'variables'

def \_\_init\_\_(self):

super(Project, self).\_\_init\_\_()

self.\_metainfo = Metainfo()

self.\_variables = VariablesContainer()

self.\_variables.insertVariable(Variable("XPROJECT", "", True, "Project",

"This variable contain current project name"))

self.\_variables.insertVariable(Variable("XPATH", "", True, "Project",

"This variable contain current project path"))

self.\_variables.forbiddenForChangeNames = ("XPROJECT", "XPATH")

@property

def metainfo(self):

return self.\_metainfo

@property

def variables(self):

return self.\_variables

@metainfo.setter

def metainfo(self, metainfo):

if not isinstance(metainfo, Metainfo):

raise TypeError("Expected type < Metainfo >")

@variables.setter

def variables(self, variables):

if not isinstance(variables, VariablesContainer):

**API.Core.Types.Variable.py**

# Variable attributes

NAME = "name"

VALUE = "value"

STATE = "state"

CATEGORY = 'category'

DESCRIPTION = "description"

class Variable(object):

def \_\_init\_\_(self, name='', value='', state=True, category='', description=''):

super(Variable, self).\_\_init\_\_()

self.\_name = name

self.\_value = value

self.\_state = state

self.\_description = description

self.\_category = category

def \_\_eq\_\_(self, var):

if not isinstance(var, Variable):

return False

#

return self.\_name == var.login

def \_\_repr\_\_(self):

return "Variable({0}, {1})".format(self.\_name, self.\_value)

def \_\_str\_\_(self):

if self.\_state:

state = "enable"

else:

state = "disable"

return '{{ {0} | {1} | {2} | "{3}" }}'.format(

self.\_name,

self.\_value,

state,

self.\_category

)

@property

def name(self): return self.\_name

@property

def value(self): return self.\_value

@property

def description(self):

return self.\_description

@property

def category(self):

return self.\_category

@property

def state(self):

return self.\_state

@property

def toDict(self):

return {

NAME: self.\_name,

VALUE: self.\_value,

DESCRIPTION: self.\_description,

CATEGORY: self.\_category,

STATE: self.\_state

}

@name.setter

def name(self, name):

if not isinstance(name, str): raise TypeError("Expected type <str>")

self.\_name = name

@value.setter

def value(self, value):

if not isinstance(value, str): raise TypeError("Expected type <str>")

self.\_value = value

@description.setter

def description(self, description):

if not isinstance(description, str): raise TypeError("Expected type <str>")

self.\_description = description

@category.setter

def category(self, category):

if not isinstance(category, str): raise TypeError("Expected type <str>")

self.\_category = category

@state.setter

def state(self, state):

if not isinstance(state, bool): raise TypeError("Expected type <bool>")

self.\_state = state

def setDict(self, dct):

name = dct[NAME]

value = dct[VALUE]

desc = dct[DESCRIPTION]

state = dct[STATE]

category = dct[CATEGORY]

#

self.\_name = name

self.\_value = value

self.\_description = desc

self.\_state = state

self.\_category= category

def setVariable(self, variable):

if isinstance(variable, Variable): raise TypeError('Expected type TypeVariable()')

self.\_name = variable.name

self.\_value = variable.value

self.\_description = variable.description

self.\_state = variable.state

def clear(self):

self.\_name = ''

self.\_value = ''

self.\_description = ''

self.\_category = ''

self.\_state = False

**API.Utility.DiGraph.py**

import time

import matplotlib.pyplot as plt

from PyQt5.QtCore import QDate

from PyQt5.QtCore import QTime

from networkx import get\_node\_attributes, draw\_networkx\_labels, shortest\_path, draw\_circular, circular\_layout

from networkx.classes import digraph

from networkx.readwrite.gpickle import read\_gpickle

from networkx.readwrite.gpickle import write\_gpickle

from networkx.readwrite.json\_graph import tree\_graph

from networkx.readwrite.json\_graph.tree import tree\_data

ERRORS = {

0: 'Success',

1: 'Saving crashed',

2: 'Crashed. Need administrator privileges',

3: 'Invalid graph type. Need DiGraph()',

4: 'Crashed. Broken file structure'

}

class \_DiGraphUtil(object):

# Name of root naode in graph

RootNode = "\_\_ROOT\_\_"

def \_\_init\_\_(self):

super(\_DiGraphUtil, self).\_\_init\_\_()

def testTree(self):

"""

root

C++

Script

Python

Script

"""

g = digraph.DiGraph()

root = self.RootNode

py\_scripts = self.uniqueName

c\_scripts = self.uniqueName

python = self.uniqueName

cpp = self.uniqueName

#

g.add\_node(root, name=self.RootNode)

g.add\_node(python, name='Python')

g.add\_node(py\_scripts, name='Scripts')

g.add\_node(c\_scripts, name='Scripts')

g.add\_node(cpp, name='C++')

#

g.add\_edge(root, python)

g.add\_edge(root, cpp)

g.add\_edge(python, py\_scripts)

g.add\_edge(cpp, c\_scripts)

#

return g

@property

def uniqueName(self):

"""

Return unique string. Correct only for GUI using.

:return:

"""

time.sleep(0.001)

d = QDate().currentDate().toString('yyyy/MM/dd')

t = QTime().currentTime().toString('hh:mm:ss:zzz')

return d + "-" + t

def DiGraph(self, \*\*attr):

"""

Return Digraph with root

Default attribute is a "name"

root ptr = self.RootNode

:return: <DiGraph>

"""

G = digraph.DiGraph()

G.add\_node(self.RootNode, name=self.RootNode, \*\*attr)

return G

def toDict(self, G: digraph.DiGraph, start\_node):

"""

Make a dict() from the DiGraph(), starting at start\_node

:param start\_node: the node from which the graph starts

:return:

"""

return tree\_data(G, start\_node)

def fromDict(self, d: dict):

return tree\_graph(d)

def shortestPath(self, G: digraph.DiGraph, src\_node, dst\_node):

return shortest\_path(G, src\_node, dst\_node)

def nodeAttributes(self, G: digraph.DiGraph, attr):

"""

Return dict{node: "attr\_value"}

"""

return get\_node\_attributes(G, attr)

def addNode(self, parent, G: digraph.DiGraph, name\_attr: str, \*\*attr):

"""

Add node with name attribute to parent node in DiGraph

:param parent: parent node

:param G: DiGraph

:param name\_attr: name attribute

:return: node pointer

"""

if not isinstance(G, digraph.DiGraph):

raise TypeError('Expected type: <DiGraph>')

#

pointer = self.uniqueName

G.add\_node(pointer, name=name\_attr, \*\*attr)

G.add\_edge(parent, pointer)

#

return pointer

def pathToRoot(self, G: digraph.DiGraph, node, attributes=[]):

"""

Return path from root to node.

If parm attributes is empty list, the path

will consist of nodes pointer.

Otherwise - dict{attr: attr\_value}

:param G: <DiGraph> tree graph

:param node: node pointer

:param attributes: list of attributes

:return: <list> path to root

"""

if not isinstance(attributes, list):

raise TypeError('Expected type <list>')

#

if not isinstance(G, digraph.DiGraph):

raise TypeError('Expected type <DiGraph>')

#

path = shortest\_path(G, self.RootNode, node)

if not attributes:

return path

else:

values = {}

for attr in attributes:

values[attr] = get\_node\_attributes(G, attr) # { atr: { ptr: value} }

#

for i, ptr in enumerate(path):

dict = {}

for attr in attributes:

dict[attr] = values[attr][ptr]

path[i] = {ptr: dict}

return path

def pathToRootAttr(self, G: digraph.DiGraph, node, attr):

"""

Return path from root to node. Instead node pointers use attribute

If parm attributes is empty list the path

will consist of nodes pointer. Otherwise - dict{attr: attr\_value}

:param G: <DiGraph> tree graph

:param node: node pointer

:param attr: attribute name

:return: <list> path to root

"""

if not isinstance(G, digraph.DiGraph):

raise TypeError('Expected type <DiGraph>')

#

if not isinstance(attr, str):

raise TypeError('Expected type <str>')

#

path = shortest\_path(G, self.RootNode, node)

values = get\_node\_attributes(G, attr) # { ptr: value }

#

for i, ptr in enumerate(path):

path[i] = values[ptr]

return path

def leaves(self, G: digraph.DiGraph):

"""

Return leaves of DiGraph

:param G:

:return: []

"""

if not isinstance(G, digraph.DiGraph):

raise TypeError('Expected type <DiGraph>')

#

return [x for x in G.nodes if G.out\_degree[x]==0 and G.in\_degree[x]==1]

def print(self, graph: DiGraph):

for node, data in graph.nodes(True):

print(node, data)

def show(self, G, attr\_name='name'):

pos = circular\_layout(G)

draw\_circular(G)

node\_labels = get\_node\_attributes(G, attr\_name)

draw\_networkx\_labels(G, pos, labels=node\_labels)

plt.show()

DIGRAPHUTIL = \_DiGraphUtil()

if \_\_name\_\_ == '\_\_main\_\_':

UT = DIGRAPHUTIL

G = UT.DiGraph()

G.add\_node(UT.RootNode, name=UT.RootNode, color='red', size='0')

p = UT.addNode(UT.RootNode, G, '1', color='red', size='0')

p = UT.addNode(UT.RootNode, G, '2', color='blue', size='0')

p = UT.addNode(UT.RootNode, G, '3', color='black', size='0')

**API.Utility.OS.py**

import json

import os

import shutil

import subprocess

from networkx import digraph

from send2trash import send2trash

from Application.API.Utility.DiGraph import DIGRAPHUTIL

class \_OSUtil(object):

MSWINDOWS = '\_\_MSWindows\_\_'

UNIX = '\_\_UNIX\_\_'

\_\_IS\_MSWINDOWS = True

\_\_IS\_UNIX = False

\_\_OS = MSWINDOWS

if os.name in ['posix', 'mac']:

\_\_OS = UNIX

\_\_IS\_UNIX = True

\_\_IS\_MSWINDOWS = False

def \_\_init\_\_(self):

super(\_OSUtil, self).\_\_init\_\_()

self.\_environs = os.environ

@property

def platform(self):

return self.\_\_OS

@property

def accessiblePlatforms(self):

return (self.MSWINDOWS, self.UNIX)

@property

def isMSWindows(self):

return self.\_\_IS\_MSWINDOWS

@property

def isUNIX(self):

return self.\_\_IS\_UNIX

@property

def systemEnvironments(self):

return self.\_environs

@property

def forbiddenSymbols(self):

return ':\*?<>|/\\'

def hasForbiddenSymbols(self, text):

"""

Return tuple of result and forbidden symbol pos.

:param text: <str>

:return: (<bool>, <int>)

"""

if not isinstance(text, str):

raise TypeError('Expected type: <str>')

#

forbidden = self.forbiddenSymbols

for i, s in enumerate(text):

if s in forbidden:

return (True, i)

return (False, -1)

def isExist(self, path):

return os.path.exists(path)

def isFile(self, path):

return os.path.isfile(path)

def isDir(self, path):

return os.path.isdir(path)

def content(self, path):

"""

Return list of files names and dirs names in directory "path"

:param path <str> directory path

:return: <list>

"""

if not self.isDir(path):

raise IsADirectoryError(path)

else:

return os.listdir(path)

def listPaths(self, path):

"""

Return list of paths in directory "path"

:param path <str> directory path

:return: <list>

"""

if not self.isDir(path):

raise NotADirectoryError("Path {0} is not a directory".format(path))

else:

paths = []

for name in os.listdir(path):

paths.append(os.path.join(path, name))

return paths

def listDirs(self, path):

"""

Return list of dirs path in directory "path"

:param path <str> directory path

:return: <list>

"""

if not self.isDir(path):

raise NotADirectoryError("Path {0} is not a directory".format(path))

else:

dirs = []

for name in os.listdir(path):

p = os.path.join(path, name)

if os.path.isdir(p):

dirs.append(p)

return dirs

def listFiles(self, path):

"""

Return list of files path in directory "path"

:param path <str> directory path

:return: <list>

"""

if not self.isDir(path):

raise NotADirectoryError("Path {0} is not a directory".format(path))

else:

files = []

for name in os.listdir(path):

p = os.path.join(path, name)

if os.path.isfile(p):

files.append(p)

return files

def isMatchSuffix(self, path, suffixes=[]):

if not isinstance(path, str):

raise TypeError('Expected type <str>')

if not isinstance(suffixes, list):

raise TypeError('Expected type <list>')

#

if not self.isFile(path):

return False

ext = self.fileExt(path)

if ext in suffixes:

return True

else:

return False

def pathFromList(self, list, exceptions=[]):

"""

Create a path from list.

For example:

list = ['media', 'images', 'london', 'me']

exceptions = ['me']

result = 'media/images/london'

:param root:

:param list:

:return:

"""

for i, name in enumerate(list):

if not isinstance(name, str):

raise TypeError('element {0} is not a <str> type'.format(str(i)))

#

path = ''

for folder in list:

if not folder in exceptions:

path = os.path.join(path, folder)

return path

def joinPaths(self, \*paths):

return os.path.join(\*paths)

def fileName(self, path):

if not self.isFile(path):

raise IsADirectoryError("Path {0} is not a file".format(path))

else:

return os.path.basename(path)

def fileTitle(self, path):

name = self.fileName(path)

splited = name.split('.')

if splited:

return splited[0]

else:

return ''

def fileExt(self, path):

name = self.fileName(path)

splited = name.split('.')

if splited:

if len(splited) > 1:

return splited[-1]

else:

return ''

else:

return ''

def isFileExt(self, path, ext):

e = self.fileExt(path)

return e == ext

def parentDirectory(self, path):

return os.path.dirname(path)

def isGoodName(self, name):

bad = self.forbiddenSymbols

for sign in name:

if sign in bad:

return False

return True

def renameFile(self, dir, old, new):

os.rename(

os.path.join(dir, old),

os.path.join(dir, new)

)

def renameDirectory(self, path, new\_name):

try:

base, name = os.path.split(path)

os.rename(

src=path,

dst=os.path.join(base, new\_name)

)

return True

except:

return False

def removeToTrash(self, path):

try:

send2trash(path)

return True

except:

return False

def removeFile(self, path):

if os.path.isfile(path):

try:

os.remove(path)

return True

except:

return False

def removeDir(self, path, ignore\_errors=False):

if os.path.isdir(path):

try:

shutil.rmtree(path, bool(ignore\_errors))

return True

except:

return False

def makeDir(self, path):

if not os.path.exists(path):

try:

os.makedirs(path)

return True

except:

return False

else:

return False

def makeTree(self, root: str, G: digraph.DiGraph):

if not isinstance(G, digraph.DiGraph):

raise TypeError('Expected type: <DiGraph>')

paths = []

node\_names = DIGRAPHUTIL.nodeAttributes(G, 'name')

#

for node in G.nodes(False):

short = DIGRAPHUTIL.shortestPath(G, DIGRAPHUTIL.RootNode, node)

path = []

for n in short:

path.append(node\_names[n])

paths.append(self.pathFromList(path))

#

paths.remove(DIGRAPHUTIL.RootNode)

for path in paths:

path = path.replace(DIGRAPHUTIL.RootNode, root)

self.makeDir(path)

def copyFile(self, src, dst):

try:

shutil.copyfile(src, dst)

return True

except:

return False

def saveToJSON(self, obj, path):

if not os.path.exists(path):

new\_file = open(path, 'w')

new\_file.close()

if os.path.exists(path):

json.dump(

obj=obj,

fp=open(path, 'w'),

indent=4

)

else:

json.dump(

obj=obj,

fp=open(path, 'w'),

indent=4

)

def loadFromJSON(self, path):

if (self.isExist(path)):

result = json.load(fp=open(path))

return result

else:

return None

def saveDiGraphToJSON(self, G: digraph.DiGraph, root\_name: str, path: str):

if not isinstance(G, digraph.DiGraph):

raise TypeError('Expected type: <DiGraph>')

#

self.saveToJSON(DIGRAPHUTIL.toDict(G, root\_name), path)

return 0

def loadDiGraphFromJSON(self, path):

"""

:return: (DiGraph, ERROR)

"""

try:

graph = DIGRAPHUTIL.fromDict(self.loadFromJSON(path))

code = 0

except:

graph = None

code = 4

return graph, code

def hideDirectory(self, path):

if os.path.isdir(path):

if self.\_\_IS\_MSWINDOWS:

cmd = """attrib +s +h "{dir}" """.format(dir=path)

os.system(cmd)

def openOSExplorer(self, path):

if os.path.isdir(path):

if self.\_\_IS\_MSWINDOWS:

path = path.replace('/', '\\')

else:

path = path.replace('\\', '/')

#

subprocess.Popen('explorer "{dir}"'.format(dir=path))

def hasEnvirons(self, \*names):

"""

Check for OS environment variable existing

:param names: env names

:return: {env: <bool>}

"""

if not isinstance(names, tuple):

raise TypeError('Expected type <list>')

for i, name in enumerate(names):

if not isinstance(name, str):

raise TypeError('Environ list has error type element({0}). Expected type <str>'.format(str(i)))

#

envs = {}

for e in names:

res = e in self.\_environs

envs[e] = res

return envs

OSUTIL = \_OSUtil()

**API.VariableProcessing.Regexps.py**

from PyQt5.QtCore import QRegExp

class RegExps(object):

def \_\_init\_\_(self):

super(RegExps, self).\_\_init\_\_()

@staticmethod

def UNIX():

rx = QRegExp('\\$\\{(.\*)\\}')

rx.setMinimal(True)

return rx

@staticmethod

def MSWindows():

rx = QRegExp('\\%(.\*)\\%')

rx.setMinimal(True)

return rx

@staticmethod

def variables():

rx = QRegExp('\\@\\{(.\*)\\}')

rx.setMinimal(True)

return rx

@staticmethod

def OS():

return RegExps.UNIX()

@staticmethod

def blended():

rx = QRegExp('\\$\\{(.\*)\\}|\\@(.\*)\\@')

rx.setMinimal(True)

return rx

**API.VariableProcessing.Parser.py**

from Application.API.VariablesProcessing.RegExps import RegExps

class Parser():

def \_\_init\_\_(self):

"""

Example: "@CGLIB@/Scripts/sceneLoad.py"

Words: @CGLIB@

Variables: CGLIB

"""

super(Parser, self).\_\_init\_\_()

def hasVariables(self, string):

"""

Checks for app variables in a string

:param string: <str> string

:return <bool>

"""

if not isinstance(string, str): raise TypeError('Excepted type: <str>')

rx = RegExps.variables()

index = rx.indexIn(string)

while index >= 0:

length = rx.matchedLength()

start = index

end = length + index

match = string[start:end]

if match: return True

index = rx.indexIn(string, index + length)

return False

def variables(self, string):

"""

Return list of app variables in string

If no app variables return []

:param string: <str> string

:return [<str>]

"""

if not self.hasVariables(string): return []

variables = []

rx = RegExps.variables()

index = rx.indexIn(string)

while index >= 0:

var = rx.cap(1)

if var: variables.append(var)

length = rx.matchedLength()

index = rx.indexIn(string, index + length)

return variables

def words(self, string):

"""

Return list of app varWords in string

If no app variables return []

:param string: <str> string

:return [<str>]

"""

if not self.hasVariables(string): return []

variables = []

rx = RegExps.variables()

index = rx.indexIn(string)

while index >= 0:

varword = rx.cap(0)

if varword: variables.append(varword)

length = rx.matchedLength()

index = rx.indexIn(string, index + length)

return variables

def variablesSlices(self, string, key='var'):

"""

if key == 'var': dict(var: slice)

return

else:

return dict(slice: var)

Indexes is tuple(start, end) of app variable position in string. (<int>, <int>)

Var is a variable

If no variables return {}

:param string: <str> string

:param key: <str> defines the key in the dict

"""

if not self.hasVariables(string): return {}

map = {}

rx = RegExps.variables()

index = rx.indexIn(string)

while index >= 0:

length = rx.matchedLength()

start = index

end = length + index

match = rx.cap(1)

if match:

if key == 'var':

map[match] = slice(start, end, 1)

else:

map[slice(start, end, 1)] = match

index = rx.indexIn(string, index + length)

return map

# OS

def hasOSVariables(self, string):

"""

Checks for OS variables in a string

:param string: <str> string

:return <bool>

"""

if not isinstance(string, str): raise TypeError('Excepted type: <str>')

rx = RegExps.OS()

index = rx.indexIn(string)

while index >= 0:

length = rx.matchedLength()

start = index

end = length + index

match = string[start:end]

if match: return True

index = rx.indexIn(string, index + length)

return False

def OSVariables(self, string):

"""

Return list of OS variables in string

If no app variables return []

:param string: <str> string

:return [<str>]

"""

if not self.hasOSVariables(string): return []

variables = []

rx = RegExps.OS()

index = rx.indexIn(string)

while index >= 0:

length = rx.matchedLength()

match = rx.cap(1)

if match: variables.append(match)

index = rx.indexIn(string, index + length)

return variables

def OSWords(self, string):

"""

Return list of app varWords in string

If no app variables return []

:param string: <str> string

:return [<str>]

"""

if not self.hasVariables(string): return []

variables = []

rx = RegExps.OS()

index = rx.indexIn(string)

while index >= 0:

varword = rx.cap(0)

if varword: variables.append(varword)

length = rx.matchedLength()

index = rx.indexIn(string, index + length)

return variables

def OSVariablesSlices(self, string, key='var'):

"""

if key == 'var': dict(var: slice)

return

else:

return dict(slice: var)

Indexes is tuple(start, end) of OS variable position in string. (<int>, <int>)

Var is a variable

If no variables return {}

:param string: <str> string

:param key: <str> defines the key in the dict

"""

if not self.hasOSVariables(string): return {}

map = {}

rx = RegExps.OS()

index = rx.indexIn(string)

while index >= 0:

length = rx.matchedLength()

start = index

end = length + index

match = rx.cap(1)

if match:

if key == 'var':

map[match] = slice(start, end, 1)

else:

map[slice(start, end, 1)] = match

index = rx.indexIn(string, index + length)

return map

**API.VariableProcessing.Processor.py**

import os

from Application.API.Utility.DiGraph import DIGRAPHUTIL as UT

from PyQt5.QtCore import QObject

from PyQt5.QtCore import QRegExp

from networkx import DiGraph

from Application.API.Core.Containers.VariablesContainer import VariablesContainer

from Application.API.Core.Exceptions import ReplicationError

from Application.API.Core.Types.Variable import NAME as AttrName

from Application.API.Utility.OS import OSUTIL

from Application.API.VariablesProcessing.Parser import Parser

from Application.API.VariablesProcessing.RegExps import RegExps

class VariablesProcessor(QObject):

"""

The class is responsible for handling the variables

and the string with the variables.

Allows you to:

- check the variable for errors; "scanError()"

- expand the variables in the string; "expandString(), expandStringOS()"

- expand the value of the variable; "expandVariable(), expandVariableOS()"

- construct a graphs for string; "stringToGraphs()"

- construct a graph for variable; "variableToGraph()"

Exceptions:

- NameError - if the invalid variable name is encountered

- ReplicationError - if the variable contains itself

(or contains variables that contain themselves or variables

by a level above)

The function "nameByError()" allows you to get the variable name by the error code.

"""

ErrorSuccess = 0

ErrorInvalid = 1

ErrorReplication = 2

ERRORS = {

ErrorSuccess: 'Success',

ErrorInvalid: 'Invalid variable',

ErrorReplication: 'Self replicationError'

}

def \_\_init\_\_(self, container: VariablesContainer, parent=None):

super(VariablesProcessor, self).\_\_init\_\_()

if parent: self.setParent(parent)

#

self.\_parser = Parser()

self.\_invalid = ''

self.\_replication = ''

self.\_container = container

def \_resetErrors(self):

self.\_invalid = ''

self.\_replication = ''

def \_makeTreeVariable(self, varword: str, graph: DiGraph, rx: QRegExp, parent=UT.RootNode):

"""

Recursively collects a tree from variables that contains a variable.

:param varword< str >: word of variable (e.g. @XLIB@)

:param graph< DIGRAPHUTIL.DiGraph(exist=True, replic=False) >: graph

:param rx< RegExps.variables() >: regular expression with variables pattern

:param parent: graph node pointer

"""

varnames = self.\_parser.variables(varword)

for name in varnames:

#{

attr\_valid = True

attr\_replic = False

# Is invalid ?

if not self.\_container.hasName(name)[0]:

self.\_invalid = name

attr\_valid = False

# Is replication ?

if name in UT.pathToRootAttr(graph, parent, 'name'):

self.\_replication = name

attr\_replic = True

# Add node to graph

p = UT.addNode(parent, graph, name, valid=attr\_valid, replic=attr\_replic)

#

expand = self.\_container.variable(name, AttrName).value

if self.\_parser.hasVariables(expand) \

and not attr\_replic \

and expand:

self.\_makeTreeVariable(expand, graph, rx, p)

#}

@property

def parser(self):

return self.\_parser

@property

def hasErrors(self):

"""

Return True if were errors after any processing

:return:

"""

return bool(self.\_invalid) or bool(self.\_violation) or bool(self.\_replication)

@property

def container(self):

return self.\_container

@container.setter

def container(self, container):

if not isinstance(container, VariablesContainer):

raise TypeError("Expected type <VariablesContainer>")

self.\_container = container

def nameByError(self, code):

"""

Return variable name by error code

:param code: <int> error code

:return: variable name

"""

if code == 1:

return self.\_invalid

elif code == 2:

return self.\_replication

else:

raise ValueError('Invalid error code. Expected 1 or 2')

def stringToGraphs(self, string):

"""

Make DiGraph for each app variable in string.

Nodes has any attributes:

"name" - <str> is name of variable

"valid" - <bool> True if exist in self.\_container, otherwise False

"replic" - <bool> True if it is self replicationError

:param string: <str> string

:return: {var\_name: <DiGraph>}

"""

if not isinstance(string, str): raise TypeError('Excepted type: <str>')

#

G = UT.DiGraph(exist=True, replic=False)

self.\_resetErrors()

trees = {}

for var in self.\_parser.variables(string):

trees[var] = self.variableToGraph(var)

return trees

def variableToGraph(self, var):

"""

Make DiGraph for app variable.

Nodes has any attributes:

"name" - <str> is name of variable

"valid" - <bool> True if exist in self.\_container, otherwise False

"replic" - <bool> True if it is self replicationError

:param string: <str> string

:return: <DiGraph>

"""

if not isinstance(var, str): raise TypeError('Excepted type: <str>')

#

G = UT.DiGraph(exist=True, replic=False)

self.\_resetErrors()

self.\_makeTreeVariable(

varword='@' + var + '@',

graph=G,

rx=RegExps.variables()

)

return G

def toContainerDepth(self, string):

"""

Return variables that are in string (depth parsing):

1 - existing <VariablesContainer>

2 - nonexistent (<str>)

:param string: <str> string with variables

:return: (<VariablesContainer>, ())

"""

if not isinstance(string, str): raise TypeError('Excepted type: <str>')

#

G = self.makeTreeString(string)

#

existing = VariablesContainer()

nonexistent = []

#

for n, d in G.nodes(True):

name = d['name']

exist, id, var = self.\_container.hasName(name)

if exist:

existing.insertVariable(var)

else:

if not name in nonexistent:

nonexistent.append(name)

return (existing, tuple(nonexistent))

def toContainer(self, string):

"""

Return variables that are in string:

1 - existing <VariablesContainer>

2 - nonexistent (<str>)

:param string: <str> string with variables

:return: (<VariablesContainer>, ())

"""

if not isinstance(string, str):

raise TypeError('Excepted type: <str>')

#

existing = VariablesContainer()

nonexistent = []

#

for name in self.\_parser.words(string):

exist, id, var = self.\_container.hasName(name)

if exist:

existing.insertVariable(vars)

else:

if not name in nonexistent:

nonexistent.append(name)

return (existing, tuple(nonexistent))

def expandVariable(self, varname):

"""

Expand app variable

:param name: <str> variable name

:return: <str> expanded string

:exception NameError

:exception ReplicationError

"""

if not isinstance(varname, str): raise TypeError('Excepted type: <str>')

#

self.\_resetErrors()

#

exist, id, variable = self.\_container.hasName(varname)

if not exist:

raise NameError('Variable {0} is not exist'.format(varname))

#

name = variable.name

expanded = [name]

rx = RegExps.variables()

string = variable.value

#

while self.\_parser.hasVariables(string):

#{

index = rx.indexIn(string)

while index >= 0:

#{

word = rx.cap(0) # $name$

name = rx.cap(1) # name

# is replication ?

if name in expanded:

#{

self.\_replication = name

raise ReplicationError('The value of the variable contains the same variable')

#}

# is valid ?

expanded.append(name)

exist, id, var = self.\_container.hasName(name)

if not exist:

#{

self.\_invalid = name

raise NameError('Variable "{0}" is not exist'.format(name))

#}

expand = self.\_container[id].value

string = string.replace(word, expand)

index = rx.indexIn(string, index + len(expand))

#}

#}

return string

def expandVariableOS(self, name):

"""

Expand OS variable

:param name: <str> os variable name

:return: <str> expanded string

"""

if not isinstance(name, str):

raise TypeError('Excepted type: <str>')

#

self.\_resetErrors()

var = OSUTIL.hasEnvirons(name)

exist = var[name]

if not exist:

raise NameError('OS variable "{}" is not exist'.format(name))

def expandString(self, string):

"""

Expand app variables in string.

:param string: <str> string

:return: <str> expanded string

"""

if not isinstance(string, str): raise TypeError('Excepted type: <str>')

#

self.\_resetErrors()

if not self.\_parser.hasVariables(string):

return string

#

rx = RegExps.variables()

s = string

index = rx.indexIn(s)

while index >= 0:

#{

word = rx.cap(0) # $name$

name = rx.cap(1) # name

expand = self.expandVariable(name)

s = s.replace(word, expand)

index = rx.indexIn(s, index + len(expand))

#}

return s

def expandStringOS(self, string):

"""

Expand OS variables in string

:param string:

:return:

"""

self.\_resetErrors()

vars = self.\_parser.OSVariables(string)

vars = OSUTIL.hasEnvirons(\*vars)

for v in vars:

#{

if not vars[v]:

raise NameError('OS variable "{0}" is not exist'.format(v))

#}

return os.path.expandvars(string)

def scanError(self, varname: str):

"""

Scans a variable for errors (depth parsing).

If the value of the variable has been completely opened,

it is checked for validity. If valid return the 0 else 2

:param varname: <str> name of variable

:return<int>: error code from ERRORS

"""

if not isinstance(varname, str): raise TypeError('Excepted type: <str>')

#

self.\_resetErrors()

#

exist, id, var = self.\_container.hasName(varname)

if not exist:

self.\_invalid = varname

return 1

#

expanded = []

rx = RegExps.variables()

string = var.value

#

while self.\_parser.hasVariables(string):

#{

index = rx.indexIn(string)

while index >= 0:

#{

word = rx.cap(0) # $name$

name = rx.cap(1) # name

# is replication ?

if name in expanded:

self.\_replication = name

return 3

# is invalid ?

exist, id = self.\_container.hasName(name)

if not exist:

self.\_invalid = name

return 1

expand = self.\_container[id].value

expanded.append(name)

string = string.replace(word, expand)

index = rx.indexIn(string, index + len(expand))

#}

#}

Техническое задание на реализацию системы

Техническое задание на реализацию системы

«CGX Studio»

**Термины и сокращения**

|  |  |  |
| --- | --- | --- |
| {Граница} | – | внешний поставщик или потребитель стрелки. Находится за рамками моделируемой системы. Название не детализируется в случаях однозначного понимания читателями диаграммы или в случаях неоднозначности. |
| {Туннель} | – | поставщик или потребитель стрелки. Название не детализируется в случаях однозначного понимания читателями диаграммы. |
| Бизнес-процесс | – | последовательность действий (подпроцессов), направленная на получение заданного результата, ценного для организации (далее Процесс). |
| Владелец процесса | – | должностное лицо, несущее ответственность за получение результата процесса и обладающее полномочиями для распоряжения ресурсами, необходимыми для выполнения процесса. |
| Входы бизнес-процесса | – | ресурсы (материальные, информационные), необходимые для выполнения и получения результата процесса, которые потребляются или преобразовываются при выполнении процесса. |
| Выходы бизнес-процесса | – | объекты (материальные или информационные), являющиеся результатом выполнения бизнес-процесса, потребляемые другими бизнес-процессами или внешними по отношению к организации клиентами. |
| Исполнитель процесса | – | подразделение или должность сотрудника, выполняющего процесс. |
| Организационная структура управления | – | совокупность специализированных функциональных подразделений, взаимосвязанных в процессе обоснования, выработки, принятия и реализации управленческих решений (далее Организационная структура). |
| Подпроцесс | – | бизнес-процесс, являющийся составной частью вышестоящего процесса. |
| Процедура | – | бизнес-процесс нижнего уровня, содержащий последовательность конечных (не требующих дополнительной детализации) операций (функций). |
| Управление бизнес-процесса | – | управляющие воздействия, регламентирующие выполнение процесса. |
| ИС | – | Информационная система |

# Введение

## Наименование Информационной системы

Полное наименование Информационной системы – CGX Studio.

## Краткая характеристика области применения

Данная система будет применятся в студиях, разрабатывающих графический контент.

CG-Студии – это предприятия, специализирующиеся на комплексном решении сложных задач в области компьютерной графики для кино и рекламы. Они сопровождают своих клиентов в течение всего цикла производства проекта: от разработки художественных решений для картины или рекламного ролика, создания аниматиков и превизуализации, супервайзинга и контроля съемки кадров под графику на площадке до постпродакшена и цветокоррекции.

Спектр решаемых студиями задач включает в себя разработку:концепт-арта;дизайна и превизуализации проекта, работу с визуальными эффектами (взрывы, разрушения, симуляции жидкостей, дымов, огня, тканей), анимацию объектов и персонажей, 2D и 3D композитинг, моушн-дизайн.

# Основание для разработки

## Основание для проведения разработки

Основанием для проведения разработки является инициатива студента.

## Наименование и условное обозначение темы разработки

Автоматизированная система организации производственного процесса в студии создания графического контента. Является децентрализованной и состоит из 5 приложений:

* «CGX Artist»;
* «CGX Supervisor»;
* «CGX Pipeline»;
* «CGX Confrence»;
* «CGX Manager».

Название системы – «CGX Studio».

# Назначение разработки

## Цели внедрения

Целями системы внедрения Информационной системы являются:

* снижение временных затрат на рутинные работы по организации и выполнению проекта;
* увеличение скорости доступа к необходимым данным при работе над проектом;
* повышение удобства работы над проектом;
* повышение скорости и удобства передачи материалов между художниками;
* снижение временных затрат на выполнение проекта

## Эксплуатационное назначение

Система предназначается для:

* обеспечения автоматизированной настройки проекта;
* обеспечения общего доступа к исходным данным;
* хранения данных проекта в виде ассетов;
* управления личной библиотекой художника;
* обеспечения внутри студийной почтовой системы;
* кастомизации рабочих мест для художников;
* автоматизации рутинных процессов.

# Требования к АИС

## Автоматизируемые процессы

**А2 Постановка задач и формирование локальных ТЗ**

**Владельцем процесса является:**

| № | Должность | Подразделение | Предмет деятельности |
| --- | --- | --- | --- |
| 1. | Менеджер | Студия |  |

**Начало выполнения процесса**

Началом выполнения процесса является сразу после формирования общего ТЗ.

**Результат процесса**

Основным результатом процесса является список задач и локальные ТЗ.

**Требования к срокам выполнения процесса**

Не более 1 часа

**Диаграмма процесса**

title

**Автоматизируемые действия**

| № | Действие | Исполнители | Используемые документы | | Требования к срокам | Функция ИС |
| --- | --- | --- | --- | --- | --- | --- |
| Входы | Выходы |
| 1. | А2.1 Постановка задач | Начальники | Общее ТЗ | Список задач | Не более 30 миут | CGX Supervisor |
| Управление задачами своего отдела |
| 2. | А2.2 Планирование задач | Начальники | Список задач | Список задач | не более 10 минут | CGX Supervisor |
| Управление задачами своего отдела |

**А3 Реализация проекта**

**Начало выполнения процесса**

Началом выполнения процесса является определяется в общем ТЗ.

**Результат процесса**

Основным результатом процесса является графический контент, тематика которого определяется в общем ТЗ.

**Требования к срокам выполнения процесса**

Определяются в общем ТЗ

**Диаграмма процесса**

title

**Автоматизируемые действия**

| № | Действие | Исполнители | Используемые документы | | Требования к срокам | Функция ИС |
| --- | --- | --- | --- | --- | --- | --- |
| Входы | Выходы |
| 1. | А3.1 Создание концепта | Концепт-художники окружений | Список задач | Концепт-арты | определяется локальным ТЗ | CGX Artist |
| Концепт-художники персонажей | ТЗ на концепты | Раскадровки | CGX Conference |
| CGX Pipeline |
| CGX Supervisor |
| 2. | А3.2 Разработка персонажей | Аниматоры | Концепт-арты | Асеты персонажей | определяется локальным ТЗ | CGX Artist |
| Сетап-художники | Раскадровки | CGX Conference |
| Текстураторы | Список задач | CGX Pipeline |
| Художник по волосам | ТЗ на персонажей | CGX Supervisor |
| Художник по тканям |
| Художник твердых тел |
| Цифровой скульптер |
| 3. | А3.3 Разработка окружений | Текстураторы | Концепт-арты | Асеты окружений | определяется локальным ТЗ | CGX Artist |
| Художник по процедурной генерации | Список задач | CGX Conference |
| Художник по процедурному моделированию | ТЗ на окружение | CGX Pipeline |
| Художник по растениям | CGX Supervisor |
| Художник твердых тел |
| 4. | А3.4 Создание визуальных эффектов | Художник по процедурной генерации | Асеты окружений | Кэш визуальных эффектов | определяется локальным ТЗ | CGX Artist |
| Художник по разрушениям | Асеты персонажей | CGX Conference |
| Художник по толпам | Концепт-арты | CGX Pipeline |
| Художник по флюидам | Список задач | CGX Supervisor |
| ТЗ на визуальные эффекты |
| 5. | А3.5 Визуализация и сборка | Монтажеры | Асеты окружений | Реализованный графический контент | определяется локальным ТЗ | CGX Artist |
| Специалист по интерактивным решениям | Асеты персонажей | CGX Conference |
| Специалист по рендерингу | Концепт-арты | CGX Pipeline |
| Художник по освещению | Кэш визуальных эффектов | CGX Supervisor |
| Художники по композитингу | Список задач |
| Художники по цветокоррекции | ТЗ на визуализацию |
| Шейдер-программист |

**А4 Отправка заказчику и доработка**

**Начало выполнения процесса**

Началом выполнения процесса является после реализации проекта.

**Результат процесса**

Основным результатом процесса является финальный продукт.

**Требования к срокам выполнения процесса**

не более 1 дня

**Диаграмма процесса**

title

**Автоматизируемые действия**

| № | Действие | Исполнители | Используемые документы | | Требования к срокам | Функция ИС |
| --- | --- | --- | --- | --- | --- | --- |
| Входы | Выходы |
| 1. | А4.1 Отправка заказчику | Менеджер | Реализованный графический контент | Список правок | не более часа | CGX Manager |
| CGX Pipeline |
| 2. | А4.2 Поиск дорабатываемых ассетов |  | Список правок |  | не более 10 минут | CGX Artist |
| CGX Pipeline |
| 3. | А4.3 Изменение ассетов | Отдел разработки | Список правок |  |  | CGX Artist |
| CGX Pipeline |
| CGX Supervisor |
| 4. | А4.4 Визуализация и сборка | Отдел композитинга | Список правок |  | определяется локальным ТЗ | CGX Artist |
| Отдел рендеринга | CGX Supervisor |
| 5. | А4.5 Отправка готового проекта | Менеджер |  |  |  | CGX Manager |
| CGX Pipeline |

## Структура Информационной системы

Информационная система должна иметь следующую структуру:

Перечень модулей и функций Информационной системы

| № | Модуль ИС | Функция ИС |
| --- | --- | --- |
| 1. | CGX Supervisor | Данная функция обеспечивает удобный, быстрый поиск и фильтрацию по ресурсам проекта. |
| Данная функция обеспечивает возможность отделам отправлять готовые результаты на конвеер |
| Данная функция дает возможность начальникам отделов отсылать файлы своим подчиненным |
| Данная функция обеспечивает возможность подключения к проекту, созданному менеджером |
| Данная функция обеспечивает начальникам отделов возможность просмотра активных художников в определенном проекте |
| Данная функция позволяет начальникам сформировать задачи проекта для своего отдела, а так же проконтролировать выполненные результаты |
| 2. | CGX Pipeline | Данная функция реализует передачу данных между начальком и сотрудниками его отдела |
| 3. | CGX Manager | Данная функция обеспечивает |
| Данная функция позволяет менеджеру просматривать асеты проекта |
| данная функция позволяет отправлять скриншоты асетов и готовые результаты заказчику |
| 4. | CGX Conference | Данная фнкция позволяет художникам организовывать локальные голосовые конференции через локальную сеть или интернет |
| Данная фнкция позволяет художникам организовывать локальные голосовые чаты через локальную сеть или интернет |
| 5. | CGX Artist | Данная функция обеспечивает автоматическое определение путевых переменных в операционной системе для дальнейшего их использования |
| Данная функция обеспечивает удобный, быстрый поиск и фильтрацию по личным и проектным ресурсам |
| Данная функция позволяет художникам брать задачи, поставленные начальником, и отправлять результаты выполнения |
| Данная функция обеспечивает художникам возможность запускать приложения с настройками, привязанными к проектоному окружению |
| Данная функция обеспечивает конвертацию .hdr и .exr файлов в .jpg формат для быстрого просмотра и при копировании помещает ссылку на .hdr/.exr файл в буфер обмена |
| Данная функция позволяет настроить "АРМ художника" |
| Данная функция обеспечивает возможность передачи файлов коллегам по своему отделу |
| Данная функция позволяет художникам подключаться к, существующему на сервере отдела, проекту |
| Данная функция позволяет художникам подключать сторонние ресурсы к проектам |
|  |

Модули и функции Информационной системы к реализации

| № | Модуль ИС | Функция ИС | Комменты |
| --- | --- | --- | --- |
| 1. | CGX Supervisor | Отсылка готового результата на главный сервер | Данная функция обеспечивает возможность отделам отправлять готовые результаты на конвеер |
| Отсылка файлов своим подчиненным | Данная функция дает возможность начальникам отделов отсылать файлы своим подчиненным |
| Подключение к существующему проекту | Данная функция обеспечивает возможность подключения к проекту, созданному менеджером |
| Просмотр активных художников | Данная функция обеспечивает начальникам отделов возможность просмотра активных художников в определенном проекте |
| Управление задачами своего отдела | Данная функция позволяет начальникам сформировать задачи проекта для своего отдела, а так же проконтролировать выполненные результаты |
| 2. | CGX Pipeline | Обеспечение связи между приложениями | Данная функция реализует передачу данных между начальком и сотрудниками его отдела |
| 3. | CGX Manager | Просмотр готовых результатов | Данная функция обеспечивает |
| Просмотр асетов | Данная функция позволяет менеджеру просматривать асеты проекта |
| Отправка данных заказчику | данная функция позволяет отправлять скриншоты асетов и готовые результаты заказчику |
| 4. | CGX Artist | Автоматическая настройка переменных проекта | Данная функция обеспечивает автоматическое определение путевых переменных в операционной системе для дальнейшего их использования |
| Быстрая фильтрация, сортировка и поиск по личным ресурсам и ресурсам проекта | Данная функция обеспечивает удобный, быстрый поиск и фильтрацию по личным и проектным ресурсам |
| Взаимодействие с задачами проекта | Данная функция позволяет художникам брать задачи, поставленные начальником, и отправлять результаты выполнения |
| Запуск приложений в окружении проекта | Данная функция обеспечивает художникам возможность запускать приложения с настройками, привязанными к проектоному окружению |
| Быстрый просмотр hdr и exr файлов | Данная функция обеспечивает конвертацию .hdr и .exr файлов в .jpg формат для быстрого просмотра и при копировании помещает ссылку на .hdr/.exr файл в буфер обмена |
| Определение пользовательских настроек | Данная функция позволяет настроить "АРМ художника" |
| Отсылка файлов коллегам по отделу | Данная функция обеспечивает возможность передачи файлов коллегам по своему отделу |
| Подключение к существующему проекту | Данная функция позволяет художникам подключаться к, существующему на сервере отдела, проекту |
| Подключение личных библиотеки к аккаунту | Данная функция позволяет художникам подключать сторонние ресурсы к проектам |

## Требования к функциональным характеристикам

### Требования к системе планирования и управления производством

Информационная система должна обеспечить планирование ресурсов предприятия и управление позаказным производством.

Требования к функциональности АИС:

1. Управление конфигурацией готовой продукции (ГП):

* Ведение нормативно-справочной информации о составе ГП с возможностью указания периода актуальности спецификации и с возможностью нахождения в производстве ГП с несколькими различными спецификациями;
* Ведение нормативно-справочной информации о технологии изготовления изделий, входящих в состав ГП с возможностью указания периода актуальности технологий и с возможностью нахождения в производстве ГП с несколькими различными технологиями;

# Управление продажами:

* Просмотр истории взаимоотношений с клиентами;
* Регистрация/корректировка заявки клиента с указанием перечня ГП, объемов, даты отгрузки, продажной цены и любых дополнительных условий;
* Просмотр актуальных экономических показателей (калькуляции) заказываемой ГП;

# Планирование производства:

* Формирование графика доступности оборудования с указанием количества доступных нормо-часов на каждый день планового периода;
* Формирование плана производства с указанием изготавливаемого изделия, его количества, используемого оборудования, подразделения на каждый день планового периода;
* Формирование плана потребности производства в материалах и комплектующих;
* Контролирование и управление загрузкой оборудования по сформированному производственному плану;
* Внесение корректировок в план производства во время его выполнения;
* План-фактный анализ плана производства;

4. Управление производством:

* Формирование сменных заданий (нарядов) на изготовление изделий;
* Назначение/переназначение нарядам исполнителей и фиксация выполнения нарядов с указанием количества выпущенных изделий, количества бракованных изделий и причин возникновения брака;
* Управление хранением и перемещением товарно-материальных ценностей (ТМЦ) в производстве;

5. Управление снабжением:

* Формирование на основании плана потребности в материалах и комплектующих заявки на покупку с указанием поставщика, номенклатуры ТМЦ, количества и сроков поставки;
* Формирование заявок на покупку на основании разовых заказов на ТМЦ от подразделений;
* Контролирование и отслеживание процесса выполнения заявок на покупку;
* Оперативный контроль остатков;
* План-фактный анализ поставок;

6. Управление затратами:

* Формирование плановой (нормативной) себестоимости ГП;
* Фиксация фактических затрат на производство;
* Расчет фактической себестоимости ГП;
* План-фактный анализ затрат.

### Требования к расчету нормативной себестоимости заказа

Нормативная себестоимость изделия и всего заказа рассчитывается по следующей методике:

1. Прямая материальная составляющая нормативной себестоимости изделия формируется на основании информации о нормативном составе этого изделия (спецификации) и установленных учетных ценах на входящие в эту спецификацию ТМЦ. Для спецификации допускается использование нескольких статей материальных затрат.

2. Величина прямой заработной платы рассчитывается на основании нормативного пооперационного состава изделия. Задаются: нормативная длительность каждой операции, профессия рабочего, требуемая для данной операции, а также разряд рабочего. Также в систему вводятся денежные расценки нормо-часов по профессиям рабочих и их разрядам.

3. Нормативная величина косвенных затрат рассчитывается как процент от задаваемой базы (величины прямых затрат по указанной статье).

Для осуществления этого расчета необходимо наличие в Информационной системе следующих данных:

1. Спецификация изготовления изделия (а также спецификации изготовления всех входящих в это изделие полуфабрикатов собственного производства);

2. Технология изготовления изделия и входящих в него полуфабрикатов: какие операции должны быть выполнены и за какое время. Кроме того, для каждой операции задаются профессия и разряд рабочего, необходимые для ее выполнения (для выпуска данного конкретного изделия);

3. Протокол учетных цен на используемые ТМЦ;

4. Денежные расценки нормо-часов для профессий и разрядов.

### Требования к расчету фактической себестоимости заказа

Фактическая себестоимость изделия и всего заказа рассчитывается по следующей методике:

1. Прямые материальные затраты на выпуск изделия рассчитываются на основании фактических данных о расходовании цехом материалов на производственные переделы. При этом сначала рассчитывается стоимость всех полуфабрикатов, входящих в данное изделие. Суммовая оценка осуществляется согласно методике, принятой в Учетной политике предприятия.

2. Заработная плата прямых производственных рабочих рассчитывается на основании данных о закрытии цеховых нарядов. В случае, если учет нарядов в ИС не ведется, заработная плата относится к прямым затратам, подлежащим распределению, т.е. распределяется на выпущенные изделия согласно некоторой базе.

3. Амортизация прямого производственного оборудования входит в состав прямых расходов в случае, если для каждого передела указывается оборудование (станок), используемое на этом переделе.

4. Прямые расходы, подлежащие распределению:

* основные материалы, расходуемые реже, чем на каждый передел (например, химикаты, норма которых на единицу продукции настолько мала, что не имеет смысл учитывать их попередельный расход даже по этой норме);
* заработная плата рабочих в случае отсутствия информации о ее попередельном распределении;
* амортизация прямого оборудования в случае наличия только ее общей месячной суммы без разбивки по переделам.

Такие расходы распределяются на выпускаемые изделия согласно выбранной базе распределения (например, пропорционально прямым материальным затратам).

1. Общепроизводственные расходы (25 счет БУ): распределяются на выпускаемые изделия пропорционально выбранной базе распределения. Доля таких расходов может оставаться или нет в составе незавершенного производства согласно принятой на предприятии Учетной политике.

2. Общехозяйственные расходы и расходы на продажу (26 и 44 счета БУ) признаются расходами текущего периода и относятся к расходам на реализацию. Распределение таких расходов на себестоимость готовой продукции можно увидеть при помощи специального отчета.

### Требования к производительности Информационной системы

Защита технических средств от воздействий электрического тока, электромагнитных полей, акустических шумов и т.п. должна осуществляться в соответствии с требованиями по эксплуатации, предъявляемыми к оборудованию его разработчиками.

Устанавливаемое оборудование должно соответствовать требованиям электробезопасности по ГОСТ Р МЭК 60065-2002.

Допустимый уровень электромагнитных полей на рабочих местах должен отвечать требованиям ГОСТ 12.1.006-84.

Факторы, оказывающие вредные воздействия на здоровье, (в том числе инфракрасное, ультрафиолетовое, рентгеновское и электромагнитное излучения, вибрация, шум, электростатические поля, ультразвук строчной частоты и т.д.), не должны превышать действующих норм (СанПиН 2.2.2./2.4.1340-03 от 03.06.2003 г. с изменениями от 25 апреля 2007 г.).

Комплекс технических средств должен соответствовать требованиям техники безопасности, основными из которых являются:

* все внешние элементы технических устройств, находящиеся под напряжением, должны иметь защитное заземление;
* технические устройства должны быть установлены в местах, обеспечивающих свободный и безопасный доступ к ним при эксплуатации и проведении профилактического обслуживания;
* сотрудники, которые работают на технических средствах, должны проходить обучение, инструктаж, проверку знаний правил, норм и инструкций по технике безопасности;
* в помещении, предназначенном для эксплуатации технических средств, должны быть обеспечены противопожарные меры безопасности.

## Требования к надежности

### Требования к обеспечению надежного (устойчивого) функционирования Информационной системы

Надежное (устойчивое) функционирование Информационной системы должно быть обеспечено выполнением Заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1. Организация бесперебойного питания технических средств;

2. Использование лицензионного программного обеспечения;

3. Регулярное выполнение рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 г. «Об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПЭВМ и оргтехники и сопровождению программных средств»;

4. Регулярное выполнение требований ГОСТ 51188-98. «Защита информации. Испытания программных средств на наличие компьютерных вирусов»;

5. Регулярное резервирование баз данных Информационной системы средствами самой Информационной системы или средствами используемой системы управления базами данных.

### Время восстановления после отказа

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать времени на перезагрузку задействованных технических и программных средств при условии соблюдения условий эксплуатации самих технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановку программных средств.

### Отказы из-за некорректных действий оператора

Отказы Информационной системы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему административных привилегий.

## . Условия эксплуатации

### Климатические условия эксплуатации

Климатические условия эксплуатации Информационной системы, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

### Требования к видам обслуживания

Информационная система не требует проведения каких-либо видов обслуживания.

### Требования к численности и квалификации персонала

Численность персонала должна быть достаточной для выполнения автоматизированных процессов в Информационной системе в полном объеме.

Для работы Информационной системы требуется персонал следующих категорий – системный администратор и конечный пользователь Информационной системы – оператор.

Системный администратор должен иметь высшее профильное образование и сертификаты компании-производителя Информационной системы. В перечень задач, выполняемых системным администратором, должны входить:

1. Задача поддержания работоспособности технических средств;

2. Задачи установки (инсталляции) и поддержания работоспособности системных программных средств;

3. Задача установки (инсталляции) Информационной системы;

4. Задачи по сопровождению и модификации Информационной системы.

Конечный пользователь (оператор) должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы. Персонал должен быть аттестован на II квалификационную группу по электробезопасности.

Необходимое количество пользователей будет определено на этапе внедрения Информационной системы.

## Требования к Информационной и программной совместимости

### Требования к информационным структурам и методам решения

Система должна быть децентрализованной, то есть состоять из:трех подсистем, одного модуля, одного самостоятельного приложения.

|  |  |
| --- | --- |
| Модуль | Пояснение |
| CGX Pipeline | Конвейер передачи данных |
| Подсистема | Пояснение |
| CGX Artist | АРМ художника |
| CGX Supervisor | АРМ начальника отдела |
| CGX Manager | Мобильное приложение менеджера |
| Приложение | Пояснение |
| CGX Conference | Приложение для голосовых конференций |

Все подсистемы и модуль являются несамостоятельными приложениями, которые должны запускаться на разных машинах. На рисунке 3.1 приведены зависимости всех компонентов системы.

### Требования к исходным кодам и языкам программирования

Система должна быть реализована на языке С++ или Python с использованием фреймворка Qt. В случае выбора языка С++, графический интерфейс должен быть написан на языке QML. Для Python требуется использование QtWidgets

### Требования к защите информации и Информационной системы

Все данные системы должны храниться на физическом носителе «Жёсткий диск», подключенном к серверному компьютеру, в структурированном виде под управлением реляционной СУБД MySQL. Исключения составляют файлы данных, предназначенные для просмотра (изображения, анимации, документы и т.п.). Такие файлы сохраняются в файловой системе, а в БД размещаются ссылки на них.

## Специальные требования

Информационная система должна обеспечивать взаимодействие с пользователем (оператором) посредством графического пользовательского интерфейса.

# Требования к программной документации

## Предварительный состав программной документации

Состав разрабатываемых программных документов должен включать в себя:

1. Техническое задание – назначение и область применения Информационной системы, технические, технико-экономические и специальные требования, предъявляемые к Информационной системе, необходимые стадии и сроки разработки, виды испытаний;

2. Пояснительная записка – схема алгоритма, общее описание алгоритма и (или) функционирования программы, а также обоснование принятых технических и технико-экономических решений;

3. Руководство оператора – сведения для обеспечения процедуры общения оператора с Информационной системой в процессе работы.

# Стадии и этапы разработки

Разработка Информационной системы осуществляется поблочно. Содержание, объемы, сроки разработки и стоимость определяется для каждого блока и оформляется отдельными дополнениями к договору. Каждый блок разрабатывается по нижеперечисленным стадиям и этапам.

## Стадии разработки

Разработка Информационной системы должна содержать стадии:

1. Техническое задание;

2. Рабочий проект;

3. Внедрение.

## Этапы разработки

На стадии разработки технического задания должны быть выполнены следующие этапы:

1. Разработка (доработка существующего) технического задания;

2. Согласование технического задания;

3. Утверждение технического задания.

На стадии рабочего проектирования должны быть выполнены следующие этапы работ:

1. Разработка Информационной системы;

2. Разработка программной документации;

3. Испытание Информационной системы.

На стадии внедрения должны быть выполнены следующие этапы разработки:

1. Тестовая эксплуатация Информационной системы;

2. Доработка Информационной системы и документации по возникшим замечаниям;

3. Промышленная эксплуатация Информационной системы.

## Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены следующие этапы работ:

1. Постановка (уточнение) задачи;

2. Определение и уточнение требований к техническим средствам;

3. Определение дополнительных требований к Информационной системе;

4. Определение стадий, этапов и сроков разработки Информационной системы и документации на нее;

5. Определение ключевых исполнителей со стороны Заказчика и закрепление их ответственности за отдельными задачами;

6. Определение потребности во внешних модулях и языков программирования для них;

7. Согласование и утверждение технического задания.

На этапе рабочего проектирования должна быть выполнена работа по программированию и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями п. 5.1 «Предварительный состав программной документации» и требованиями ГОСТ 19.101-77 Единая система программной документации. Виды программ и программных документов.

На этапе испытаний Информационная система проходит два вида испытаний:

1. Внутренние испытания при сдаче блока Информационной системы в опытную эксплуатацию силами Разработчика. При этом должны быть выполнены перечисленные ниже виды работ:

* разработка, согласование и утверждение порядка и методики испытаний;
* проведение испытаний;
* фиксация результатов испытаний;
* корректировка Информационной системы и программной документации по результатам испытаний.

2. Испытания Информационной системы в ходе тестовой и промышленной эксплуатации силами Заказчика. При этом должны быть выполнены перечисленные ниже виды работ:

* определение и локализация дефекта в Информационной системе;
* внесение дефекта в реестр дефектов с его подробным описанием;
* устранение дефекта, корректировка Информационной системы и программной документации по результатам тестовой и промышленной эксплуатации.

На этапе тестовой и промышленной эксплуатации Информационной системы должны быть выполнены следующие работы:

1. Выделение автоматизированных рабочих мест пользователей Информационной системы, закрепление за ними функционала и соответствующих руководств оператора;

2. Работы по подготовке и передаче Информационной системы и документации в эксплуатацию в подразделениях Заказчика;

3. Обучение пользователей правилам работы с Информационной системой с фиксацией результатов обучения;

4. Консультирование пользователей по возникающим вопросам, связанным с работой Информационной системы;

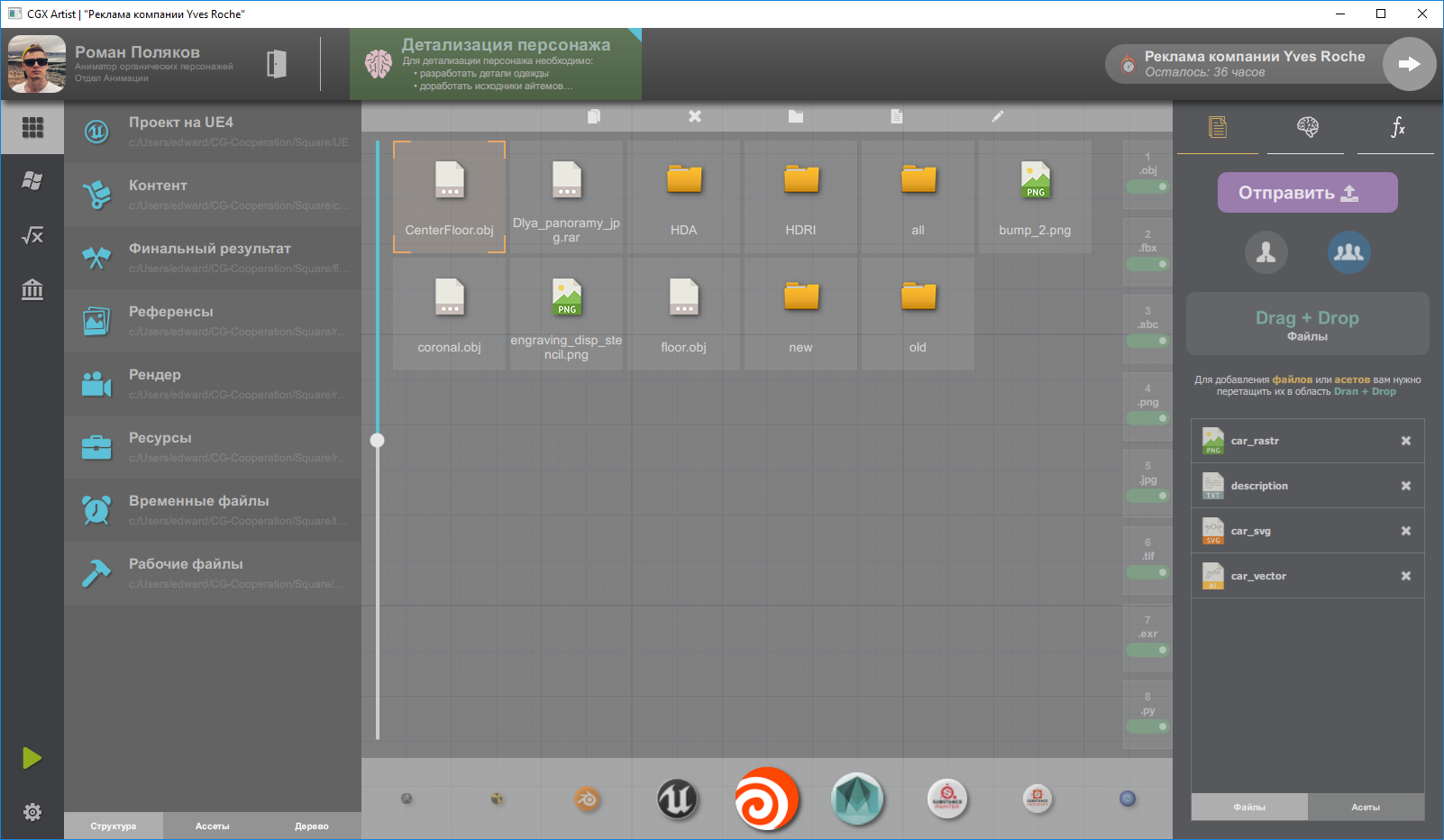
5. Методическая помощь при решении вопросов учета в Информационной системе.

7. Порядок контроля и приемки

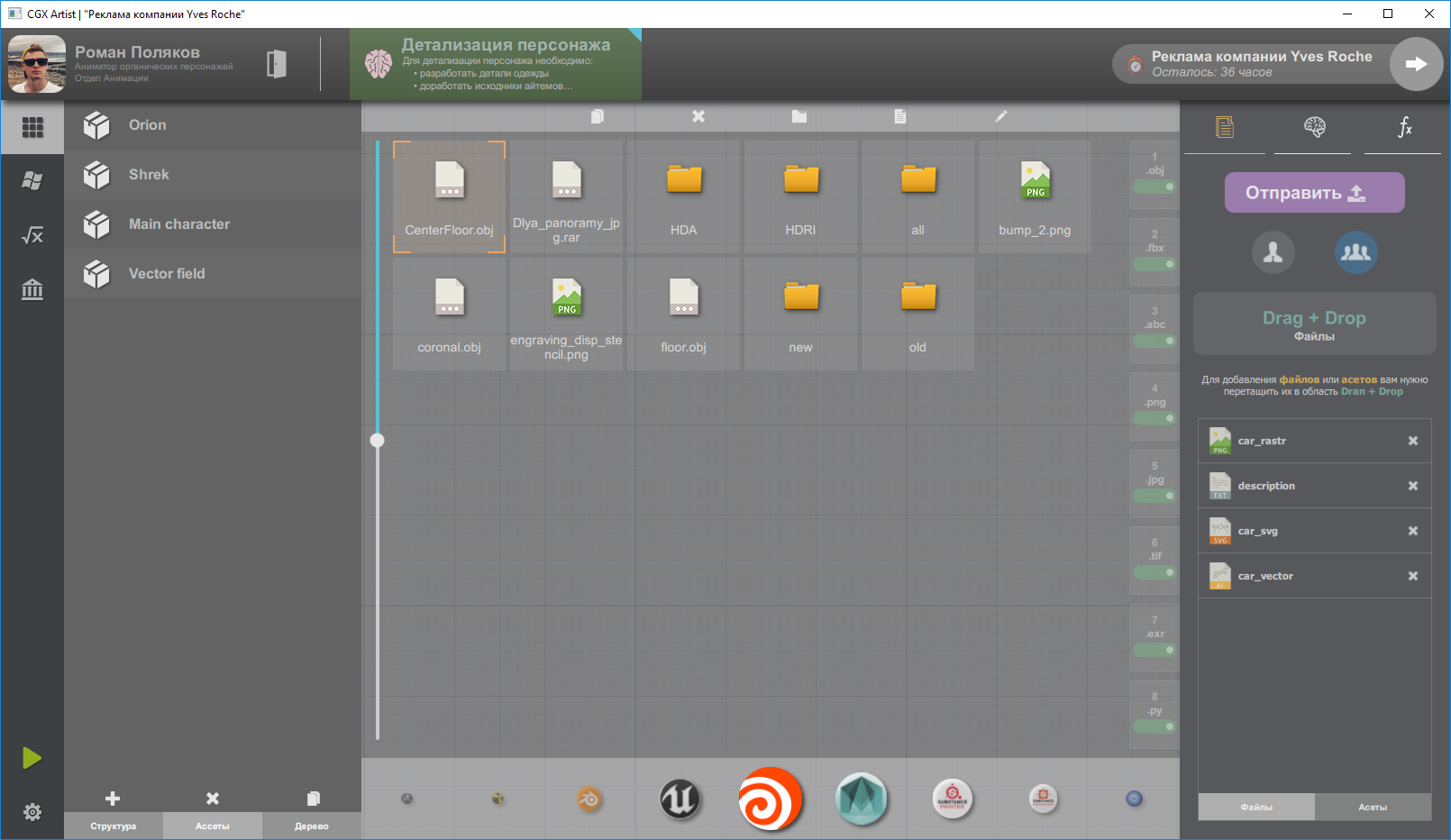
После того как каждый блок прошел этап промышленной эксплуатации Информационной системы, оформляется «Акт сдачи-приемки системы», который утверждается должностными лицами сторон, подписавшими Договор на разработку Информационной системы, или лицами, ими уполномоченными.

1. Примеры графического интерфейса системы

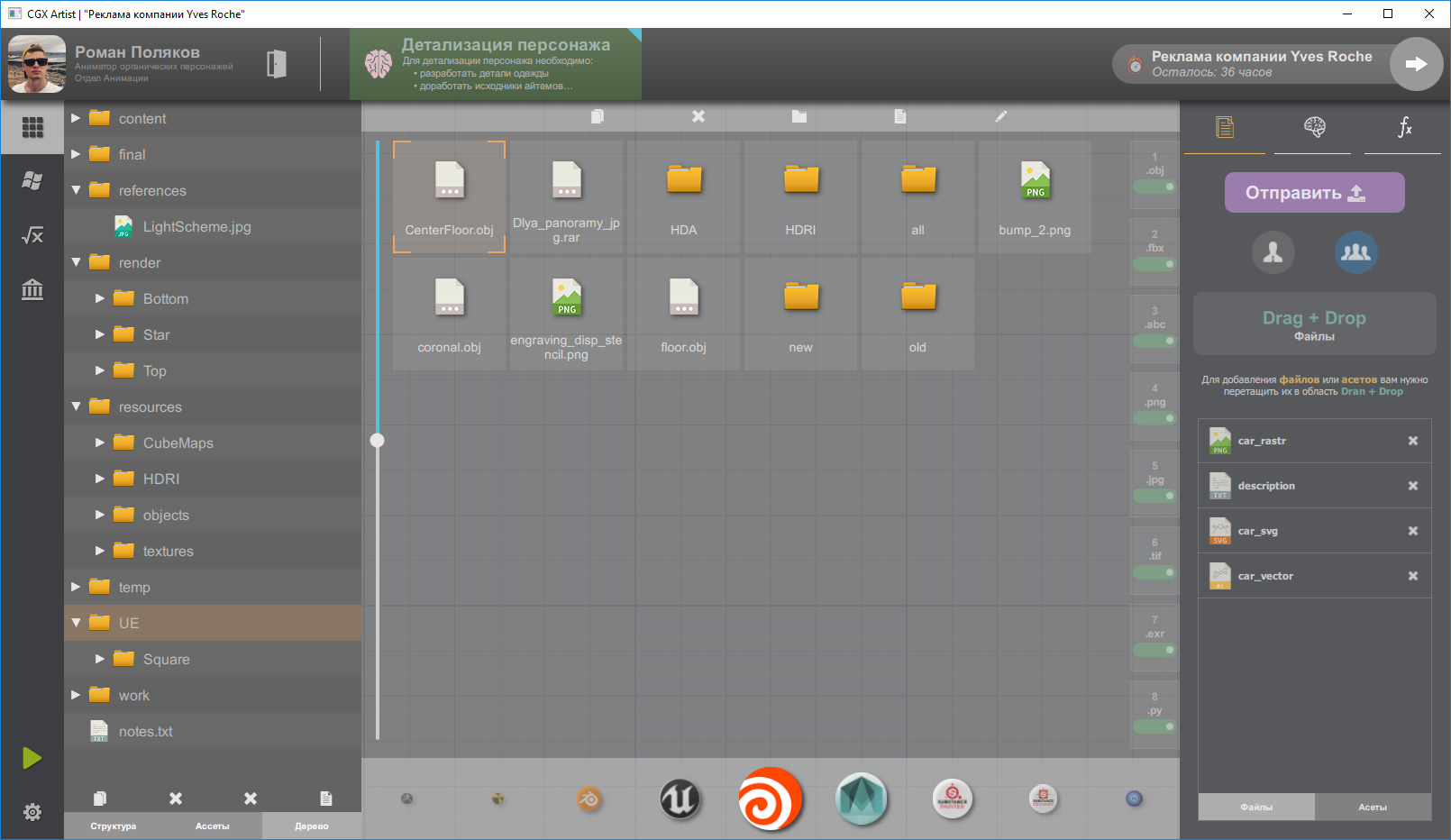
**Примеры графического интерфейса системы «CGX Studio»**



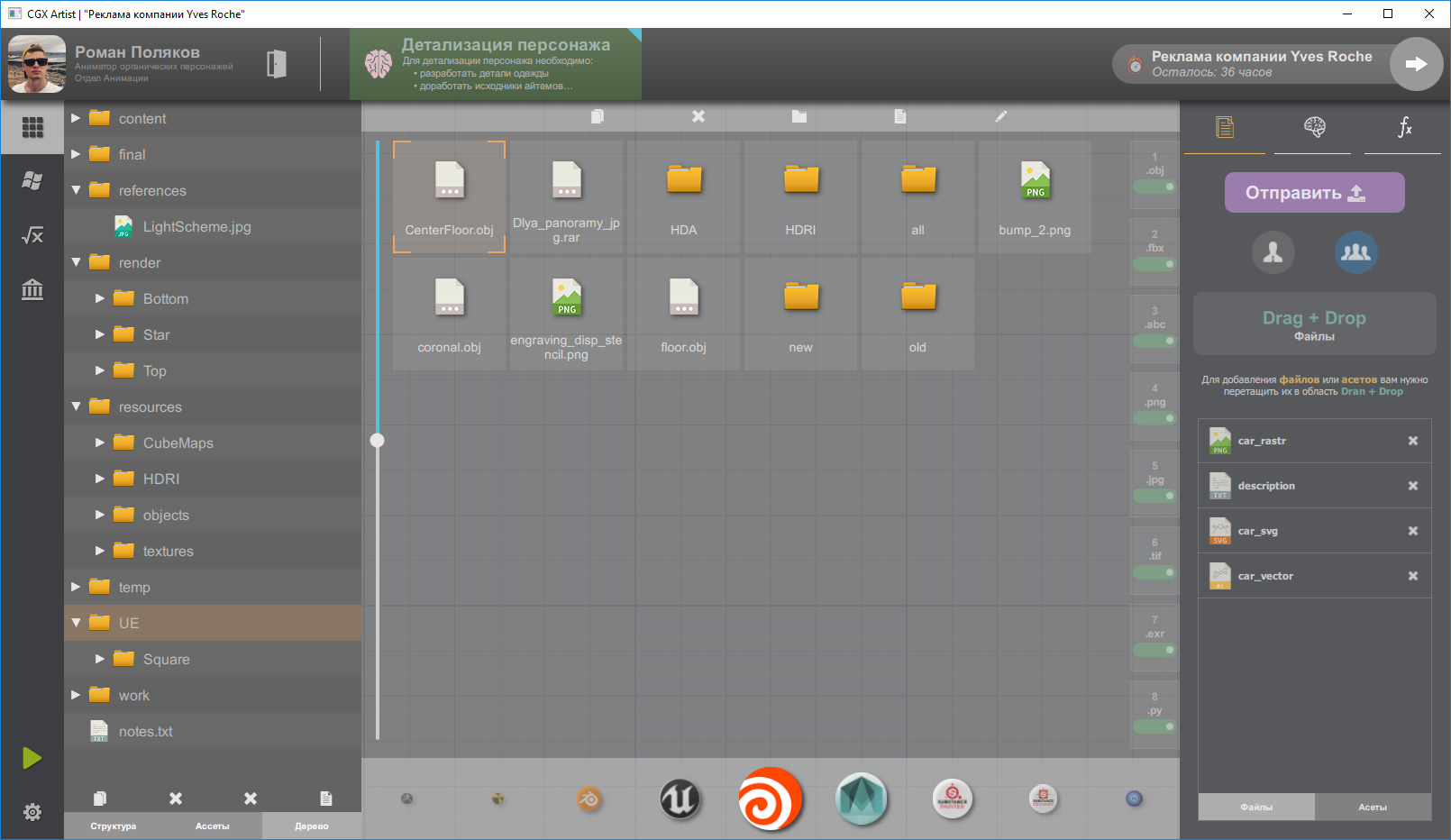
*Рис. 1 – Видеокадр «CGX Artist. Главное окно приложения»*



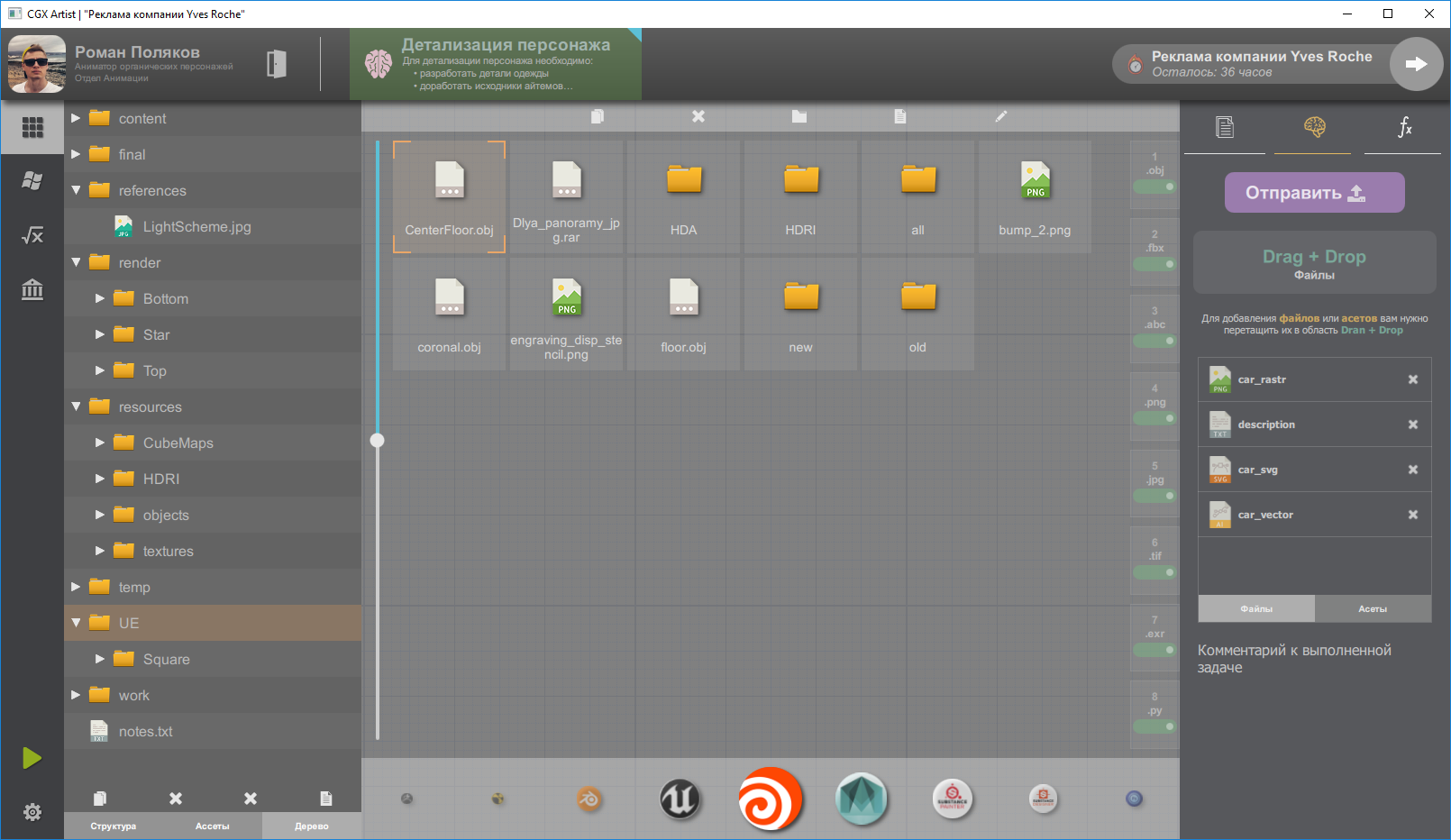
*Рис. 2 – Видеокадр «CGX Artist. Ассетная структура проекта»*



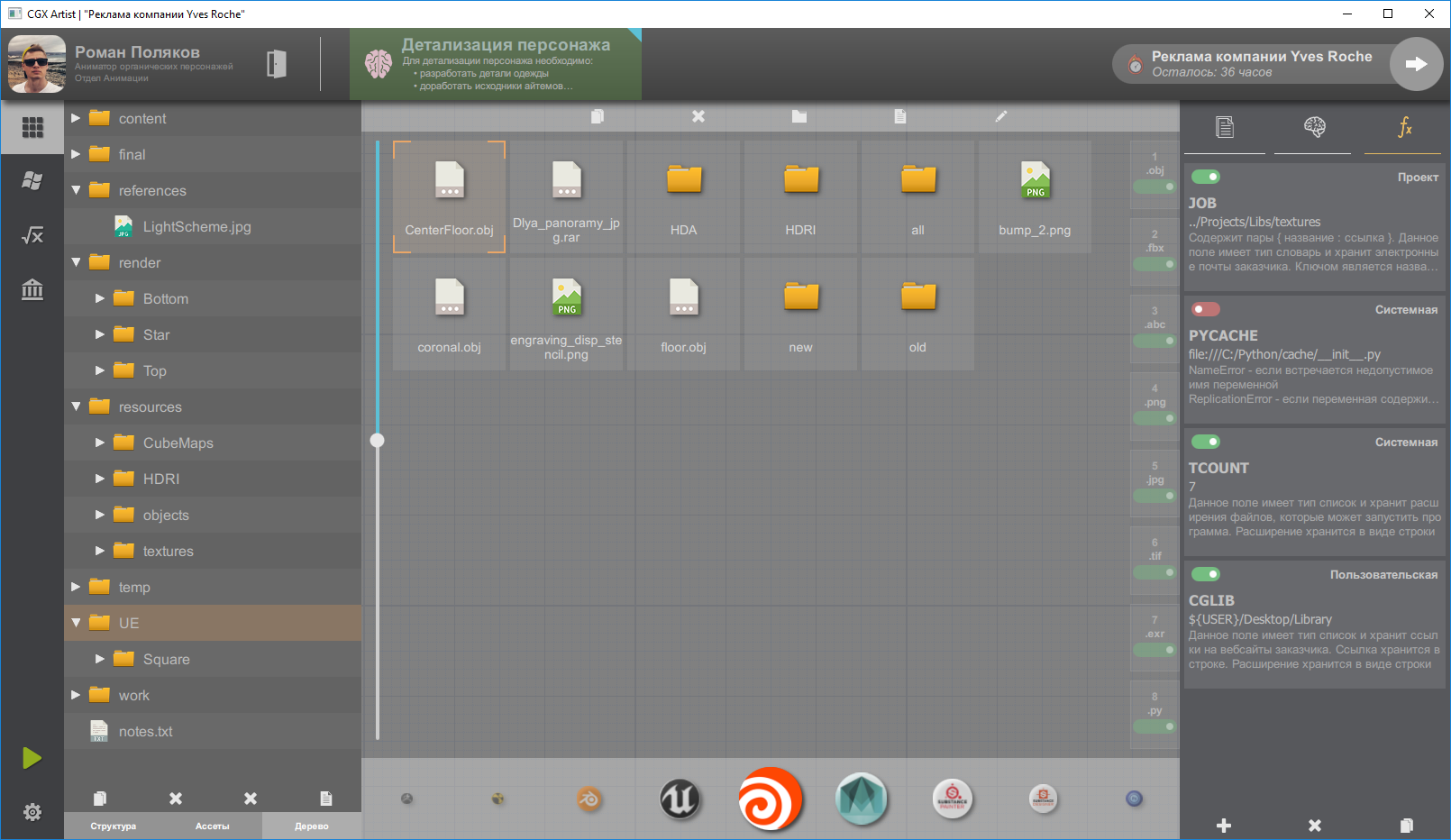
*Рис. 3 – Видеокадр «CGX Artist. Древовидная структура проекта»*



*Рис. 4 – Видеокадр «CGX Artist. Отправка файлов»*



*Рис. 5 – Видеокадр «CGX Artist. Отправка отчета по решенной задаче»*



*Рис. 5 – Видеокадр «CGX Artist. Переменные проекта»*