Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Институт повышения квалификации
и переподготовки

Кафедра «Профессиональная переподготока»

В. А. Бельский

АНИМАЦИОННАЯ ГРАФИКА

ПОСОБИЕ
по одноименной дисциплине
для слушателей специальности переподготовки
1-40 01 77 «Web-дизайн и компьютерная графика»
заочной формы обучения

Гомель 2019

УДК 004.928(075.8)
ББК 32.973я73
Б44

Рекомендовано кафедрой «Профессиональная переподготовка» ИПКиП ГГТУ им. П. О. Сухого
(*протокол № 4 от 26.12.2018 г.*)

Рецензент: профессор каф. «Информатика» ГГТУ им. П. О. Сухого
д-р физ.-мат. наук *В. П. Кудин*

Бельский, В. А.

Б44 Анимационная графика : пособие по одноим. дисциплине для слушателей специальности переподготовки 1-40 01 74 «Web-дизайн и компьютерная графика» заоч. формы обучения / В. А. Бельский. – Гомель : ГГТУ им. П. О. Сухого, 2019. – 92 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://elib.gstu.by. – Загл. с титул. экрана.

Рассматриваются виды анимации и основные приемы для создания анимации. Приводится подробный обзор программных средств для создания мультипликационных фильмов. Отдельные главы посвящены созданию анимированных элементов при веб-разработке.

УДК 004.928(075.8)
ББК 32.973я73

СОДЕРЖАНИЕ

Лекция №1. История анимации 4

Лекция №2. Виды анимации 16

Лекция №3. Обзор программ для создания компьютерной анимации 28

Лекция №4. Основы CSS-анимации 53

Лекция №5. Анимация с использованием API Canvas 78

Лекция №1. История анимации

Анимация способна объяснить всё, что может породить человеческий разум. (Уолт Дисней)

Введение

Анимация (animation) - от латинского "anima" - душа, анимация означает одушевление или оживление.

Мультипликация, анимация, мультипликационное кино, анимационное кино - вид киноискусства, произведения которого создаются методом покадровой съёмки последовательных фаз движения рисованных (графическая или рисованная мультипликация) или объёмных (объёмная или кукольная мультипликация) объектов.

Искусством мультипликации занимаются мультипликаторы (аниматоры).

Первые попытки запечатлеть движение в рисунках относят к палеолитическим пещерным рисункам, где животных изображали с множеством ног, перекрывающих друг друга.

В Шахр-и Сохта (Иран) был найден глиняный сосуд, возраст которого оценивается в 5000 лет. На стенках сосуда сделано 5 изображений козла в движении.
Также были найдены рисунки, в Египте (относят к 2000 году до н.э.). Были заявления о том, что эти рисунки необходимо назвать первыми примерами анимации, однако, это не совсем корректно, так как не было оборудования, способного показать эти рисунки в движении.

Мультипликацию считают ветвью киноиндустрии с момента ее появления. Однако это интересное направление можно также успешно связать с живописью и графикой. Талант художника плюс технические возможности – и на свет рождается искусство, которое не оставляет равнодушным ни детей, ни взрослых.

Корни возникновения мультипликации связаны со стробоскопом – оптической игрушкой, изобретенной бельгийским

изобретателем Жозефом Плато в 1832 году. Принцип этого устройства был прост – на край круга наносился цикличный рисунок. Например, бегущая лошадь, которую изображали несколько раз в разных стадиях движения. При вращении круга рисунок сливался, и возникала иллюзия движущегося объекта.

Первым настоящим мультипликатором принято считать француза Эмиля Рейно. Он создал аппарат праксиноскоп, который состоял из крутящегося барабана, системы зеркал и фонаря. В 1892 году Рейно запустил своеобразный аттракцион – оптический театр. Там он демонстрировал зрителям комические сюжеты продолжительностью 15-20 минут. Это случилось за несколько лет до знаменитой премьеры братьев Люмьер, то есть мультипликация стала известна французам даже несколько раньше, чем кинофильмы.

Рисунок 1.1 – Прасиноскоп Эмиля Рейно

Первенство

На почетное звание претендуют сразу несколько работ. Больше всех отличился американский художник, мастер карикатуры Джеймс Стюарт Блэктон. Целых три его произведения фигурируют среди номинантов. В 1900 создан короткий немой мультфильм «Очарованный рисунок». В 1906 году им представлена работа «Смешные переходы забавных лиц». Она демонстрировала быструю смену гримас, отражающих разнообразные человеческие эмоции. В этой картине сочетались техника кино и искусство графики. Интересно, что в американских кинозалах показ сопровождался игрой пианиста, а порой даже целого оркестра.

Рисунок 1.2 – Кадр из мультфильма «Смешные переходы забавных лиц»

В следующем году Блэктон выпустил «Отель с призраками». Эта работа была более совершенной в техническом и анимационном плане. Но этот мультфильм был кукольным, поэтому многие специалисты оспаривают его право на звание первой детской анимации.

Претендует на первое место и европейский мультфильм. Эмиль Коль, художник-карикатурист из Франции, создал работу под названием «Фантасмагория» (1908 год). Особенность этой черно-белой ленты – наличие четкого сюжета. Еще одно отличие заключалось в том, что у ключевого персонажа ярко выражены индивидуальные черты характера. Для создания этого двухминутного мультфильма Эмиль Коль сделал чуть менее 700 рисунков.

Рисунок 1.3 – Кадр из мультфильма «Фантасмагория»

Появление объемной мультипликации

Первой работой, выполненной в виде объемной мультипликации, стала «Прекрасная Люканида, или Война усачей с рогачами». Ее создатель – Владислав Старевич, российский ученый-биолог. Автор воплотил работу в жизнь в 1910, публика же смогла увидеть анимацию лишь в 1912.

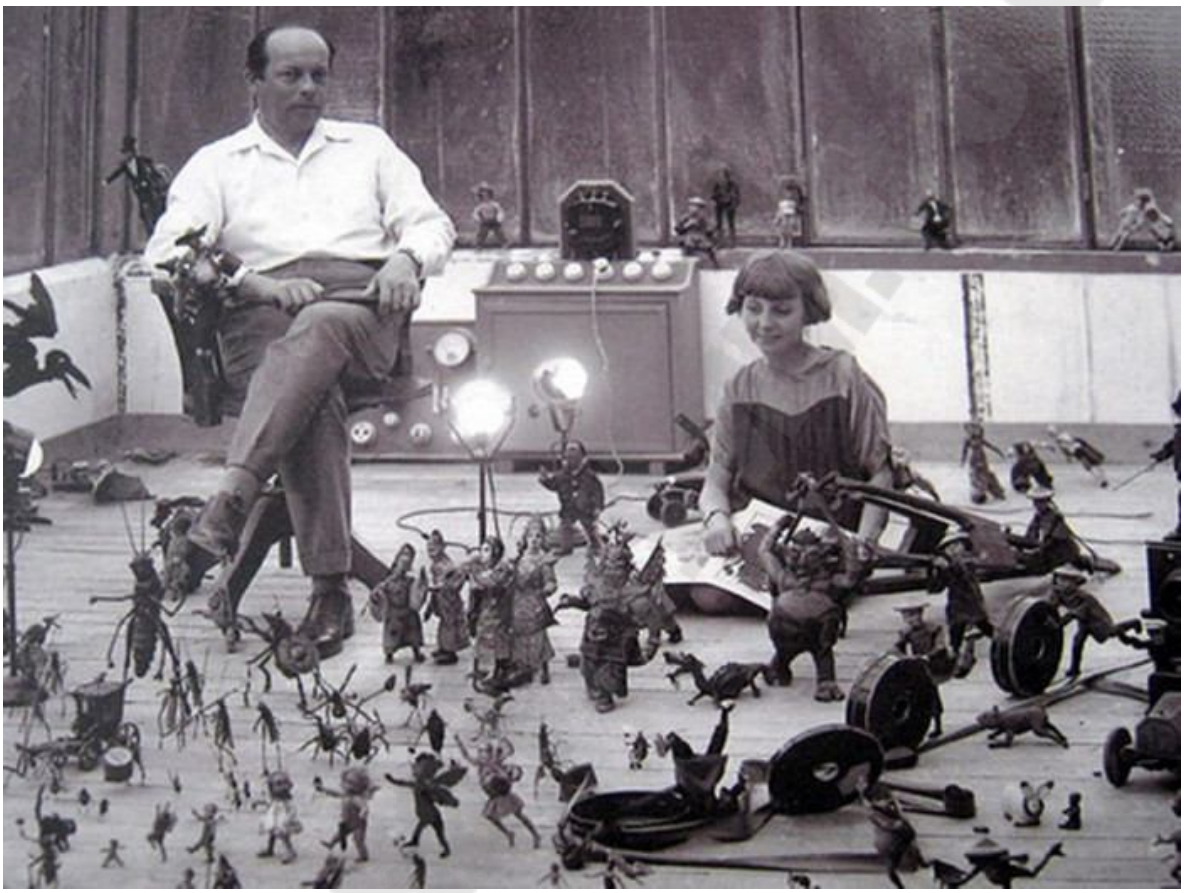


Рисунок 1.4 – Съемка мультфильма «Прекрасная Люканида, или Война усачей с рогачами»

Интересно, что изначально ученый планировал снять научную картину о драке двух жуков-рогачей. Но здесь его постигла неудача. Тогда ему пришла в голову мысль использовать препарированных насекомых. Прикрепив к ним тонкие незаметные глазу проволоки, он кадр за кадром отснял сотни движений. Таким образом, появился мультфильм, который имел логично построенный сюжет - завязку, развитие и кульминацию.

Старевич заслуженно закрепил за собой звание создателя первого объемного мультфильма, став при этом еще и основоположником

этого вида искусства в России. Сам же биолог стал в дальнейшем работать в кукольной анимации.

Работы Уолта Диснея

Уолт Дисней, пожалуй, самая известная личность в истории мультипликации. На его работах уже выросли несколько поколений детей, и непременно вырастут еще. Интересный сюжет, привлекательные герои, обилие ярких оттенков по-настоящему завораживают. Глядя на динамичные, красочные мультфильмы Диснея, сложно поверить, что первые его работы были монохромными – «Смехограмма» (1921 год), «День Алисы на море» (1924 год), «Удачный кролик Освальд».

Рисунок 1.5 – Кадр из мультфильма «День Алисы на море»

А вот известный по всему миру герой Уолта Диснея Микки Маус впервые мелькнул в кадре в далеком 1928 году в «Безумном самолете», следом он стал действующим лицом «Пароходика Вилли», который был первой лентой со звуковым сопровождением.

Первые среди полнометражных

Появление «полного метра» в мультипликации тоже весьма значимое событие. Самым первым в мире полнометражным мультиком считается «Апостол», вышедший в 1918. Его создал мультипликатор и режиссер из Аргентины Квирини Кристиани.

Рисунок 1.6 – Кадр из мультфильма «Белоснежка и семь гномов»

Ранней, но уже более известной широкой публике, полнометражной анимацией стала лента Уолта Диснея «Белоснежка и семь гномов». Выйдя в 1977, она принесла своему автору настоящую славу.

Советская мультипликация

Детская анимация в России изначально была кукольной. Первой считают мультипликационную работу, созданную балетмейстером Александром Ширяевым в 1906 году. В ней на фоне статичных декораций 12 силуэтов выполняют связки движений.

Следующим мультфильмом, который был предназначен не только для детей – «Советские игрушки». Он был придуман и воплощен в жизнь документалистом Давидом Кауфман. В нем красноармейцы украшали елку фигурками своих врагов. После него вышла це-

лая серия советских мультфильмов, которые были направлены на пропаганду патриотизма и рассказывали о ключевых моментах жизни страны.

Рисунок 1.7 – Кадр из мультфильма «Советские игрушки»

Первый полнометражным мультиком в СССР стал «Новый Гулливер», вышедший в 1935 году.

Советская графическая мультипликация возникла в 1924-1925 годах. Первые фильмы были сделаны художником А. Бушкиным. Художники-мультипликаторы А. Бушкин и А.Г. Иванов для создания некоторых фильмов пользовались весьма несложной, но выразительной техникой плоских марионеток. Из плотной бумаги или картона вырезали плоские марионетки, в местах сочленений их скрепляли шарнирами. Марионетка накладывалась либо на стекло съёмочного стола, за которым находился рисованный фон или панорама, либо непосредственно на рисованную декорацию − место действия для персонажей фильма.

В 1936 году в Москве по решению правительства была создана специальная студия рисованных фильмов "Союзмультфильм". В этот период художники-мультипликаторы в своих работах постепенно начинают осваивать цвет (выходят первые цветные детские фильмы "Сладкий пирог" (1937, реж. Д. Бабиченко), "Красная шапочка" (1937, реж. В. и З. Брумберг), "Маленький Мук" (1938, реж. О. Ходатаева) и мн. др.). С 1969 года на "Союзмультфильме" А. Котеночкин создаёт многосерийный фильм "Ну, погоди!"

Появление цветных мультфильмов и 3D-анимации

Первый в мире цветной мультфильм в 1932 году выпустила корпорация «Walt Disney Studios». Им стала короткометражная работа «Цветы и деревья». Примечательно, что эта анимация изначально была отснята в черно-белой. Но потом гениальный мультипликатор пошел на риск и впервые решил добавить цвета, применив трехцветную технологию «Триколор».

Рисунок 1.8 – Кадр из мультфильма «Цветы и деревья»

Первый российский цветной мультфильм датирован 1936 годом, им стал «Лиса и волк». Его автор – художница Сарра Мокиль.

В 1995 году вышла первая мировая анимация формата 3D - полнометражная «История игрушек». Российский мультфильм такого формата вышел в 2010 году – «Белка и Стрелка. Звездные собаки».

Развитие не стоит на месте. За время своего существования мультипликация превратилась из примитивных, сменяющих друг друга рисунков в полноцветные анимационные фильмы формата 3D, насыщенные спецэффектами и компьютерной графикой.

Компьютерная анимация

Наиболее ранние компьютерные анимации конца 60- начала 70-х годов были получены совместными усилиями исследователей в университетских лабораториях и отдельных художников. Первые исследования в области компьютерной графики и анимации проводились в 1963 году в MIT (Массачусетском технологическом институте), когда Иван Сазерленд разработал диалоговую систему решения задач с ограничениями на векторном дисплее.

Первые работы по компьютерной анимации в СССР связаны с именем Юрия Баяковского. В 1990 году на конференции SIGGRAPH ассоциация ACM присвоила ему титул "Computer Graphics Pioneer". Сейчас Юрий Матвеевич возглавляет в МГУ Лабораторию компьютерной графики и мультимедиа на факультете ВМК (graphics.cs.msu.su), но продолжает работать и в Институте прикладной математики им. Келдыша РАН, где он много лет заведовал отделом компьютерной графики и где еще в 1964 году в соавторстве с физиком Тамарой Сушкевич выполнил первую в СССР работу по "машинной графике". Это была как раз анимация, небольшой фильм об обтекании цилиндра плазмой. В 1971 году в ИПМ был разработан более совершенный софт для создания компьютерных фильмов и установлена камера для съемки кадров с экрана дисплея. При помощи этой системы вскоре были сделаны еще две впечатляющие анимации. Одна визуализировала движение робота, другая – смоделированное "приливное взаимодействие" двух галактик.

В начале 70-х несколько значительных работ по анимации были произведены в штате Юта: анимированная рука и лицо, созданные Эдвин Кэтмулом (1972), идущая и говорящая фигура, сделанная Берри Веслером в 1973, говорящее лицо, созданное Фредом Парком в

1974. По нынешним меркам качество изображения анимации было примитивным, однако для той поры это были впечатляющие результаты.

В конце 70х Нью-Йоркский технологический институт начал работу над проектом создания фильма «The works», полностью созданным с помощью компьютера с использованием трехмерной компьютерной анимации. Проект не был закончен, но отдельные фрагменты показывались на конференциях SIGGRAPH. Эти фрагменты демонстрировали высокое качество визуализации, сочлененные фигуры и взаимодействующие объекты. При создании «The works» использовалась система BBOP – трехмерная система анимации сочлененных фигур.

В начале 80х годов вклад в развитие анимации внесла лаборатория Даниэля Толман и Нади Магнинат-Толман (компьютерные анимации «Dream Flight», «Tony de Peltrie», «Rendez – vous a Montrual»). Среди других, внесших вклад в этот период были Эд Эмшвиллер, который показал двигающиеся текстурные карты в Sunstone; Джим Блинн, который создал анимацию «Voyager»; Дон Гринберг, создавший архитектурные обходы университетского городка Корнельского университета и другие.

К 1980-м годам заметно продвинулось аппаратное обеспечение. В это же время графические программы становились все более сложными: Тернер Виттед ввел понятие трассировки лучей с устранением погрешностей дискретизации; Нельсон Макс создал несколько фильмов о молекулах и один из первых фильмов с анимированными волнами; Лорен Карпентер создал полет вокруг фрактального ландшафта и т.д.

Очень важным направлением в компьютерной анимации является создание искусственного человека, неотличимого от настоящего. В этих фильмах качество анимации человека было таким, что для кинозрителей было очевидно, что персонаж нереален, и в этих фильмах компьютерные актеры играли второстепенные роли. Последние же достижения в моделях освещения и текстурирования позволяют много более реалистично моделировать человека.

Фильм «Young Sherlock Holmes» (1986) был первым, который поместил искусственный персонаж в художественный фильм.

Начинают выходить фильмы, в которых все чаще используются компьютерные спецэффекты: имитация торнадо системой частиц, превращения в вампиров, летающие персонажи, персонажи без ног и т.д. 1993 год – выход «Jurassic Park» с анимированными моделями динозавров, 1995 год – выход «Jumanji» с моделями как реальных, так и выдуманных животных.

Рисунок 1.9 – Аватар - компьютерная анимация и технология производства анимации - motion capture

В конце 20 века появились новая технология производства анимации - захват движения (motion capture). Объект захвата движения обычно оборудуется какой-либо измерительной аппаратурой так, чтобы положение ключевых точек на нем было легко обнаружить и зафиксировать в любой момент времени. После этого в эти положения можно поместить трехмерную компьютерную модель и анимировать ее так, что она будет повторять движения рассчитанных положений.

Метод захвата движения применяется в производстве CGI-мультфильмов, а также для создания спецэффектов в фильмах. Широко используется в игровой индустрии. С использованием этого метода в 2004 году созданы мультфильмы «Полярный экспресс» (модель – Том Хэнкс), «Последняя фантазия», «Властелин колец» (модель – Энди Серкис). В 2006 - 2012 году при помощи этой технологии был создан «Ренессанс», «Беовульф», «Рождественская история», «Аватар», «Гарри Поттер», «Пираты Карибского моря», «Ранго».

Все же в конце этой лекции вернемся к Эмилю Рейно.

В 1893–1894 он создал свой шедевр «Вокруг кабинки», но уже в 1895 рождение кинематографа нанесло ему сокрушительный удар: рукотворные ленты Рейно не могли соперничать с более быстрыми в производстве и более дешёвыми кинофильмами. Отчаявшийся изобретатель разбил свой аппарат и утопил его в Сене вместе с лентами, частично уцелели всего две из них, «Бедный Пьеро» и «Вокруг кабинки».

Заслуги Рейно перед анимацией переоценить трудно, но его нельзя назвать создателем первого в истории мультфильма, поскольку в производстве его фильмов не использовалась киноплёнка как носитель изображения.

В честь первого показа "светящихся пантомим" Рейно в 1892 году 28 октября отмечается как Международный день анимации. Праздник был утверждён АСИФА в 2002 году.

Рисунок 1.10 – Эмиль Рейно во время показа «Вокруг кабинки»

Лекция №2. Виды анимации

"Здесь, тем не менее, мы оглядываемся назад, но не стоим на месте. Мы стремимся вперед, открываем новые пути, беремся за новые дела, потому что мы любопытны... и любопытство гонит нас по новым дорогам. Только вперед." (Уолт Дисней)

На протяжении всего своего существования человек пытался отразить движение в своем искусстве. Первые попытки передачи движения в рисунке относятся примерно к 2000 году до нашей эры (Египет).

Еще один пример движения найден в пещерах Северной Испании: это рисунок кабана с восьмью ногами.

Сегодня передача движения может быть реализована средствами анимации.

Анимацией называется искусственное представление движения в кино, на телевидении или в компьютерной графике путем отображения последовательности рисунков или кадров с частотой, при которой обеспечивается целостное зрительное восприятие образов.

Анимация, в отличие от видео, использующего непрерывное движение, использует множество независимых рисунков.

Синоним «анимации» – «мультипликация» – очень широко распространен в нашей стране. Анимация и мультипликация – это лишь разные определения одного и того же вида искусства.

Более привычный для нас термин произошел от латинского слова «мульти» – много и соответствует традиционной технологии размножения рисунка, ведь для того, чтобы герой «ожил», нужно мно-

гократно повторить его движение: от 10 до 30 рисованных кадров в секунду.

Принятое в мире профессиональное определение «анимация» (в переводе с латинского «анима» – душа, «анимация» – оживление, одушевление) как нельзя более точно отражает все современные технические и художественные возможности анимационного кино, ведь мастера анимации не просто оживляют своих героев, а вкладывают в их создание частичку своей души.

О технологии анимации

Для начала, что бы не запутаться, хотелось бы выделить 3 основных критерия по которым можно судить об анимации: виды анимации, методы анимации и стили анимации. Для многих – это одно и тоже, но это не совсем так. Виды анимации – это то, в каком виде или форме демонстрируется мультфильм (рисованный, кукольный и т. д). Методы анимации - это технические особенности с помощью которых создается анимация (покадровая анимация, программируемая анимация и т. д). И наконец, стиль анимации – это художественный прием, который используется в анимации (реализм, аниме и т. д). В этой лекции мы рассмотрим понятие "виды анимации" в более развернутом виде, а впоследствии прибавим к этому "методы" и "стили". Все это вместе и есть технология анимации.

1) **Рисованная классическая анимация.**

Один из самых интересных и распространенных видов анимации, это так называемая классическая анимация. Классическую анимацию делают рисуя на прозрачной пленке (или кальке) каждый отдельный кадр. Затем эти кадры собирают в специальной программе монтажа. Такая анимация очень живая, плавная, пространственная, но дорогая. Примером такой анимации могут послужить мультфильмы студии "Дисней" и "Союзмультфильм".

Если анимация создаётся исключительно по рисункам на бумаге, каждый элемент изображения нужно повторять на всех рисунках. Чтобы сократить огромный объём работы, были разработаны альтернативные методы анимации. Наиболее известным и самым распространённым из них был метод, при котором

использовалась *келевая анимация.* Его суть заключается в том, что элементы сцены, которые могут двигаться (например, Гомер Симпсон), изображаются на прозрачном материале, именуемом келем, и накладываются на изображённый отдельно фон (возможно, гостиную Симпсонов).

Рисунок 2.1 – Кадр из мультфильма «*Белоснежка и семь гномов» студии "Дисней"*

2) **Перекладная анимация.**

Старейший вид анимации. Суть этого вида анимации в том, что нарисованный на картоне или бумаге объект режется на отдельные кусочки и эти кусочки передвигаются (перекладываются) от кадра к кадру. Отсюда и название – перекладка (в англоязычной литературе – cut-out animation). Многие считают такую анимацию примитивной, но в умелых руках такие мультфильмы могут получиться очень интересными. Одним из самых ярких примеров перекладной анимации можно смело назвать мультфильм "Ежик в Тумане" режиссера Юрия Норштейна. "Ежик в тумане" был признан мировым сообществом одним из лучших мультфильмов всех времен.

Рисунок 2.2 – Кадр из мультфильма «Ежик в тумане» режиссера Юрия Норштейна

3) **Живопись на стекле.**

Суть такой анимации в рисовании масляными красками по стеклу. Каждый кадр при этом – это живописная картина, которая видоизменяется мазками художника. Ярким примером такой анимации является произведение Александра Петрова "Старик и море", которое было удостоено премией "Оскар".

Рисунок 2.3 – Кадр из мультфильма фильма "Старик и море" режиссера Александра Петрова

4) **Кукольная анимация.**

Тоже старинный вид анимации. Все куклы и декорации в кукольной анимации изготавливаются вручную, что делает ее такой же дорогой, как и классическая анимация. Тем не менее такой вид анимации очень популярен даже сегодня (несмотря на распространение компьютерной 3d-анимации).

Рисунок 2.4 – Кадр из мультфильма фильма "Кошмар перед рождеством " режиссера Тома Бертона

5) **Пластилиновая анимация**

Название пластилиновая анимация говорит само за себя. Добавлю только то, что пластилиновая анимация вышла из кукольной анимации и стала популярной у нас после появления мультфильмов "Падал прошлогодний снег" режиссера Александра Татарского.

6) **Компьютерная 2d-анимация.**

Хотим мы того или не хотим, но на смену старым видам анимации приходят новые. Компьютерную 2д анимацию почему-то принято называть Flash анимацией и это не совсем верно. Flash - это только одна программа. Можно назвать и другие мощные компьютерные программы для создания 2д анимации, например: After Effect, Anime Studio Pro, Toon Boom Studio и еще множество платных и бесплатных программ. В настоящее время компьютерной 2д анимацией занимаются почти все студии. Именно такую анимацию

мы видим сегодня в сериалах по телевизору, в интернете, в казуальных компьютерных играх и т. д.

Рисунок 2.5 – Кадр из мультфильма фильма "Падал прошлогодний снег " режиссера Александра Татарского

Рисунок 2.6 – Кадр из анимационного сериала "Симсоны"

7) **3d-анимация.**

3d-анимация – вид мультипликации созданный на базе компьютерных 3d-программ. Это самый молодой и самый

перспективны вид анимации. С развитием компьютерных технологий стало возможным не только рисовать графику и анимацию в двухмерной плоскости (2d-анимация), но и оживлять трехмерные формы. Технология эта очень сложная, но если рассказать коротко, то процесс производства выглядит примерно так. Сначала рисуются концепт арты (любым способом), по этим рисункам создают трехмерную геометрию модели. Затем текстуры, которые надеваются на форму персонажа или объекта. Далее создают кости объекта и прикрепляют их к форме, что бы форма смогла двигаться. Аниматоры получают подготовленную модель для движений и начинают ее оживлять. Чем лучше подготовлена модель, тем ее движения получаются естественнее и пластичнее. После создания анимации сцена визуализируется (переводится из 3d в обычную картинку). Яркими примерами такой анимации являются мультфильмы студии Пиксар.

Рисунок 2.7 – Кадр из мультфильма "Рататуй" студии Пиксар

8) **Комбинированная анимация.**

Комбинированная анимация – это совмещение любого из видов анимации с видеофильмом. Ранними примерами такой анимации могут быть: фильм "Кто подставил кролика Роджера", мультфильм "Приключение Капитана Врунгеля" и т. д. С развитием 3d технологии и компьютерных спецэффектов этот вид анимации встречается в художественных фильмах все чаще и чаще. Основной особенностью

современной комбинированной анимация заключается в ее полной реалистичности.

Рисунок 2.8 – Кадр из *фильма* "Мир юрского периода" режиссера Стивена Спилберга

9) **Другие виды анимации.**

Кроме перечисленных видов анимации существуют и другие, например: песочная анимация, лазерная анимация, фото анимация, игольчатая анимация и т. д. Однако, эти виды считаются менее популярными и в данной лекции мы позволим себе о них лишь упомянуть.

Технологии создания анимации

В настоящее время существует различные технологии создания анимации:

1. **Классическая (традиционная) анимация** представляет собой поочередную смену рисунков, каждый из которых нарисован отдельно. Это очень трудоемкий процесс, так как аниматорам приходится отдельно создавать каждый кадр.
2. **Стоп-кадровая (кукольная) анимация**. Размещенные в пространстве объекты фиксируются кадром, после чего их положение изменяется и вновь фиксируется.
3. **Спрайтовая анимация** реализуется при помощи языка программирования.

4. **Морфинг** – преобразование одного объекта в другой за счет генерации заданного количества промежуточных кадров.
5. **Цветовая анимация** – при ней изменяется лишь цвет, а не положение объекта.
6. **3D-анимация** создается при помощи специальных программ (например, 3D MAX). Картинки получаются путем визуализации сцены, а каждая сцена представляет собой набор объектов, источников света, текстур.
7. **Захват движения (Motion Capture) –** первое направление анимации, которое дает возможность передавать естественные, реалистичные движения в реальном времени. Датчики прикрепляются на живого актера в тех местах, которые будут приведены в соответствие с контрольными точками компьютерной модели для ввода и оцифровки движения. Координаты актера и его ориентация в пространстве передаются графической станции, и анимационные модели оживают.

Принципы анимации

При создании анимационных фильмов используются некоторые общие принципы. Большинство из них сформулировано для анимации Диснея и первоначально относилось к мультфильмам, выполненным в технике традиционной анимации, но практически все они применимы и при других технологиях.

Вот основные из них:

1. **«Сжатие и растяжение»** *(squash & stretch)*. Этот принцип произвел революцию в мире анимации. Суть принципа состоит в том, что живое тело всегда сжимается и растягивается во время движения. Перед прыжком персонаж сжимается как пружина, а в прыжке наоборот растянут. Главным правилом при этом является постоянный объем - если персонаж растянули (stretch - деформация по оси Y), то он обязательно должен быть сжат для сохранения объема своего тела (squash - деформация по оси X).
2. **«Подготовительное действие»** *(Anticipation)*. В реальной жизни для произведения какого-либо действия, человеку часто приходится делать подготовительные движения. Например, перед прыжком человеку необходимо присесть, для того чтобы бросить что-либо руку необходимо завести назад. Такие действия

называются отказными движениями, т.к. перед тем как сделать что-то персонаж как бы отказывается от действия. Такое движение подготавливает зрителя к последующему действию персонажа и придает инерцию движениям.

3. **Сценичность** *(staging)*. Для правильного восприятия персонажа зрителями все его движения, позы и выражения лица должны быть предельно просты и выразительны. Этот принцип основан на главном правиле театра. Камера должна быть расположена так, чтобы зритель видел все движения персонажа.
4. **«Ключевые кадры»** *(Pose to Pose)*. До открытия этого принципа движения рисовались, и поэтому результат было трудно предсказать, т.к. сам художник еще не знал, что он нарисует. Этот принцип предусматривает предварительную компоновку движений - художник рисует основные моменты и располагает персонажа на сцене, а уж потом ассистенты прорисовывают все кадры движения. Этот подход резко увеличил производительность, т.к. заранее планировались все движения, и результат был именно таким как задумывалось. Но чтобы создать какое-то конкретное движение, была необходима тщательная проработка каждого «кусочка». Разрабатывая выразительные позы художник вкладывает все свое мастерство, поэтому именно эти моменты должны быть дольше видны зрителю. Для этого ассистенты дорисовывают движения так, что больше всего кадров оказывается рядом с ключевыми позами. При этом персонаж как бы проскальзывает движение от одной компоновки к другой, медленно выходя из позы и замедляясь у другой.
5. **«Сквозное движение и захлест»** *(follow through / Overlapping actions)*.

   Суть принципа состоит в том, что движение никогда не должно прекращаться. Существуют такие элементы как уши, хвосты, одежда, которые постоянно должны находиться в движении. «Сквозное движение» обеспечивает непрерывность движения и плавность перехода фаз, например, из бега в шаг и наоборот. Движение отдельных элементов тела, в то время как тело уже не двигается, называется захлестом. Захлест выражается в сценах смены фаз движения. Если персонаж резко тормозит после бега, мягкие части тела не могут остановиться вместе с жесткими и происходит небольшой захлест (волосы, уши, хвосты и т. д). При ходьбе движение начинается с бедер, а уж потом распро-

страняется до лодыжек. Таким образом, все движения персонажа связаны в отдельную цепочку, и появляется возможность жестко описать правила, по которым он двигается. Движение, при котором один элемент следует за другим, называется сквозным движением.

6. **«Движения по дугам»** *(arcs)*. Живые организмы всегда передвигаются по дугообразным траекториям. До этого применялся метод прямолинейного движения, в связи с чем, движения выглядели механическими - как у роботов. Характер траектории зависит, как правило, от скорости движения. Если персонаж движется резко, траектория распрямляется, если же медленно, то траектория еще больше загибается.
7. **Второстепенные действия** *(Secondary actions)*. Часто для придания персонажу большей выразительности используют вторичные движения. Они служат для того, чтобы акцентировать внимание на чем-нибудь. Например, горюющий персонаж может часто сморкаться в платок, а удивленный подергивать плечами. Вторичные действия получили широкое распространение в мировой анимации. Благодаря их использованию персонажи становятся более живыми и эмоциональными.
8. **Расчет времени** *(Timing)*. Этот принцип позволяет придать персонажу вес и настроение. Как зритель оценивает вес персонажей? Вес персонажа складывается из таких факторов как скорость перемещения и инертность. Для того чтобы персонаж двигался в соответствии со своим весом, художник рассчитывает время движения и захлеста для каждого персонажа. При расчете времени учитываются вес, инертность, объем и эмоциональное состояние героя. Настроение также передается скоростью движений персонажа. Так подавленный персонаж движется очень вяло, а воодушевленный достаточно энергично.
9. **Преувеличение** *(Exaggerrate and Caricature)*. Уолт Дисней всегда требовал от своих работников большего реализма, на самом деле стремясь больше к "карикатурному реализму". Если персонаж должен был быть печальным, он требовал, чтобы его делали мрачным, счастливого же нужно было делать ослепительно сияющим. С помощью преувеличения увеличивается эмоциональное воздействие на зрителей, однако, персонаж приобретает карикатурный характер.

10. **Профессиональный рисунок**. Рисунок основа всего. На студии Диснея довольно часто встречаются таблички вроде: "Чувствуется ли в твоем рисунке вес, глубина и равновесие?". Принцип профессионального рисунка также воспрещает рисовать "близнецов". "Близнецами" называют любые элементы рисунка, которые повторяются дважды или являются симметричными "Близнецы" очень часто появляются помимо воли художника, сам не замечая того, он рисует две руки в одном и том же положении.
11. **Привлекательность** *(Appeal)*. Привлекательность персонажа – путь к успеху всего фильма. Как же определить, привлекателен ли персонаж? Привлекательным может быть любой предмет, если смотришь на него с удовольствием, обнаруживая в нем простоту, обаяние, хороший дизайн, очарование и магнетизм. От привлекательного персонажа невозможно оторвать взгляд. Даже самый противный герой фильма должен быть привлекательным, чтобы удержать зрителей у экрана.

Лекция №3. Обзор программ для создания компьютерной анимации

1) Vectorian Giotto – бесплатная

Назначение программы - создание 2D-мультипликации формата Flash. Vectorian Giotto - упрощенная версия известной программы Adobe Flash.

Нельзя требовать от бесплатной программы той же функциональности, что и от больших и дорогих программ для профессионалов по созданию Flash. Но Vectorian Giotto с успехом может применяться для создания анимированных баннеров и других flash-элементов для вставки на веб-страницы.

Программа Vectorian Giotto проста в изучении и применении.

Имеется:

- комплект инструментов для векторного рисования
- большой набор эффектов (более 50)
- использование графика времени, слоев, градиентов, настройки скорости анимации, подбор фона и пр.
- использование функции tween - создание ключевых кадров, все промежуточные кадры программа создает самостоятельно.
- операции с объектами - перемещение, вращение, наклон, изменение размеров, цвета.
- вставка текста
- возможность вставки звука
- шаблоны баннеров
- поддержка вставки своего кода ActionScript 2.
- встроенный предварительный просмотр.

2) Pencil2d – бесплатная

Pencil является кросс-платформенной программой с открытым исходным кодом для создания 2D анимации. Основное назначение – сделать классическую анимацию (мультфильмы и т.д.). Pencil анимация – графическое ПО для Mac OS X, Windows и Linux.

Программа дает возможность создания традиционной рисованной анимации (мультфильмов) с применением векторной и растровой графики. Pencil является абсолютно бесплатным, и распространяется как программа с открытым исходным кодом.

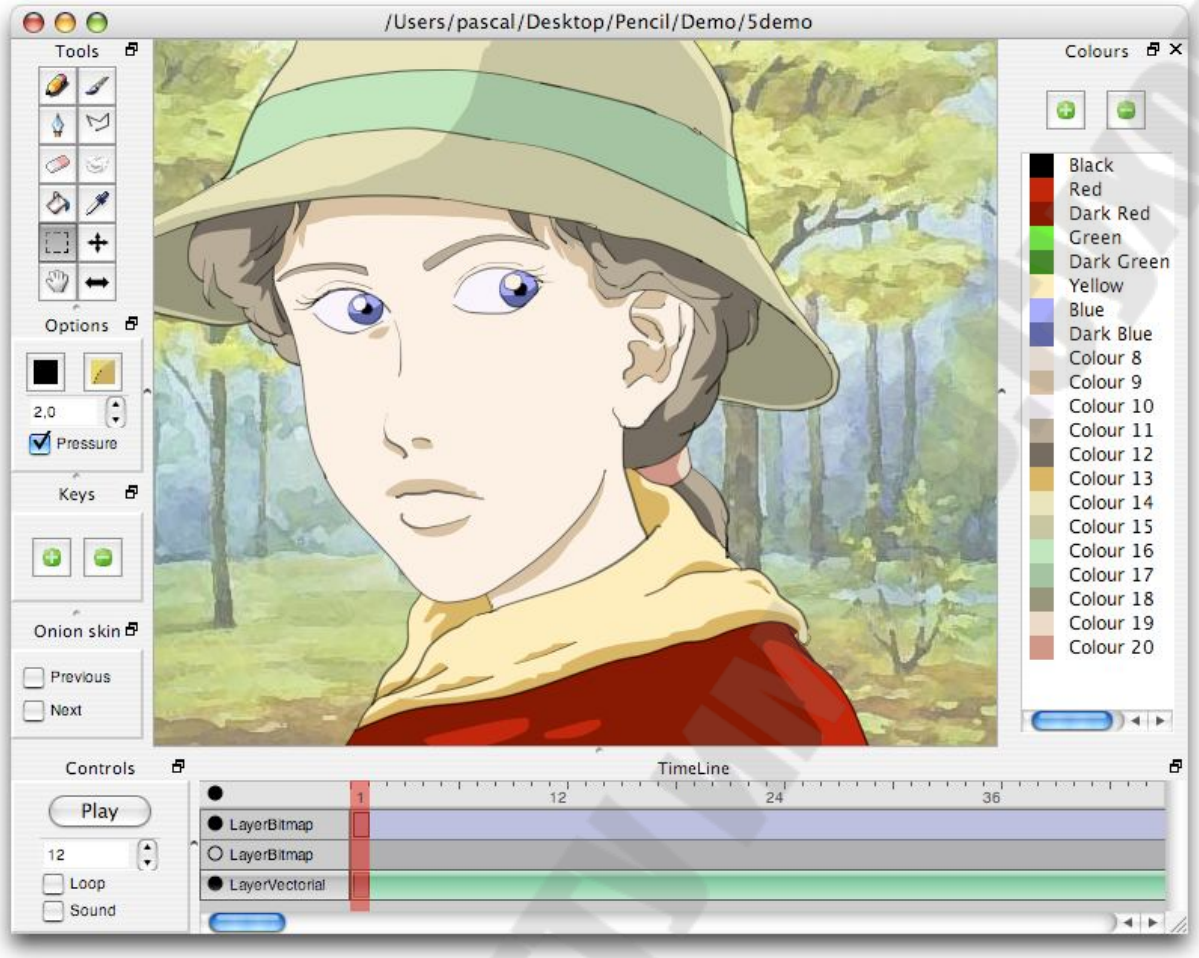


Рисунок 3.1 – Рабочее окно программы Pencil2d

Pencil не предназначен для веб-ориентированной векторной анимации на основе программ, такие как Flash. Его главной целью является создание традиционной анимации. Ни он не пытается конкурировать с коммерческим программным обеспечением таргетинга профессиональной сфере анимации. Pencil предназначен для простой программы, **давая возможность каждому сделать 2D анимацию**.

3) VPaint – векторный редактор для 2D-анимации

Программа позволяет создавать независимые от разрешения иллюстрации и анимации используя инновационные методы.
Команда VPaint гордится тем, что она является программным обеспечением с открытым исходным кодом, лицензированным по лицензии MIT

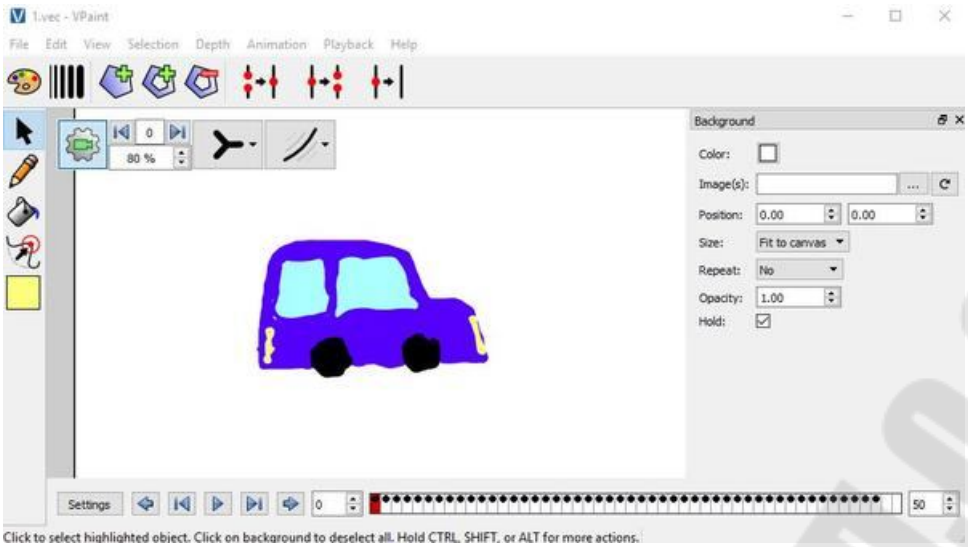


Рисунок 3.2 – Рабочее окно программы VPaint

4) Tupi – Создание 2D анимации

Несмотря на то, что она все еще находится в ранней стадии разработки, включает в себя множество функции:

- Модульный дружественный интерфейс;
- Анимация и модули предварительного просмотра;
- Основные инструменты для векторного иллюстрирования;
- Временная шкала и лист экспозиции;
- Импорт растровых изображений;
- Экспорт в различные форматов (Ogg Theora, AVI, MPEG, SWF и последовательность PNG изображений).

5) Synfig Studio – Создание 2D анимации

Программа для 2D анимации, выполненная в виде мощного решения для создания анимации с кинематографическим качеством с применением векторных и растровых картинок. Она избавляет от необходимости создавать анимацию кадр за кадром, что в свою очередь, дает возможность создания 2D анимации более высокого качества с меньшим количеством ресурсов. Synfig Studio доступна для Windows, Linux и MacOS X.

Особенности:

- множество элементов векторной и все слои генерируются параметрически, следовательно, даже при смене целевого разрешения проекта, единственная пикселизация будет происходить в

импортированных растровых изображениях, а не во встроенных компонентах;
- независимое временное разрешение;
- независимое разрешение пространства;
- анимация ключевых кадров автоматически подбирает интерполяцию, в результате плавного движения;
- широкий динамический диапазон изображений (HDRI);
- при использовании плавающей точки в расчетах изображений, обработка HDRI позволяет полотну получить гораздо более широкий диапазон яркости пикселей, в результате чего лучше световые эффекты, и улучшенная цветовая композиция;
- ориентирована на художника;
- слои;
- Synfig поддерживает множество различных типов слоев, градиентов, фильтров, искажений, трансформаций, фракталов;
- и многое другое.

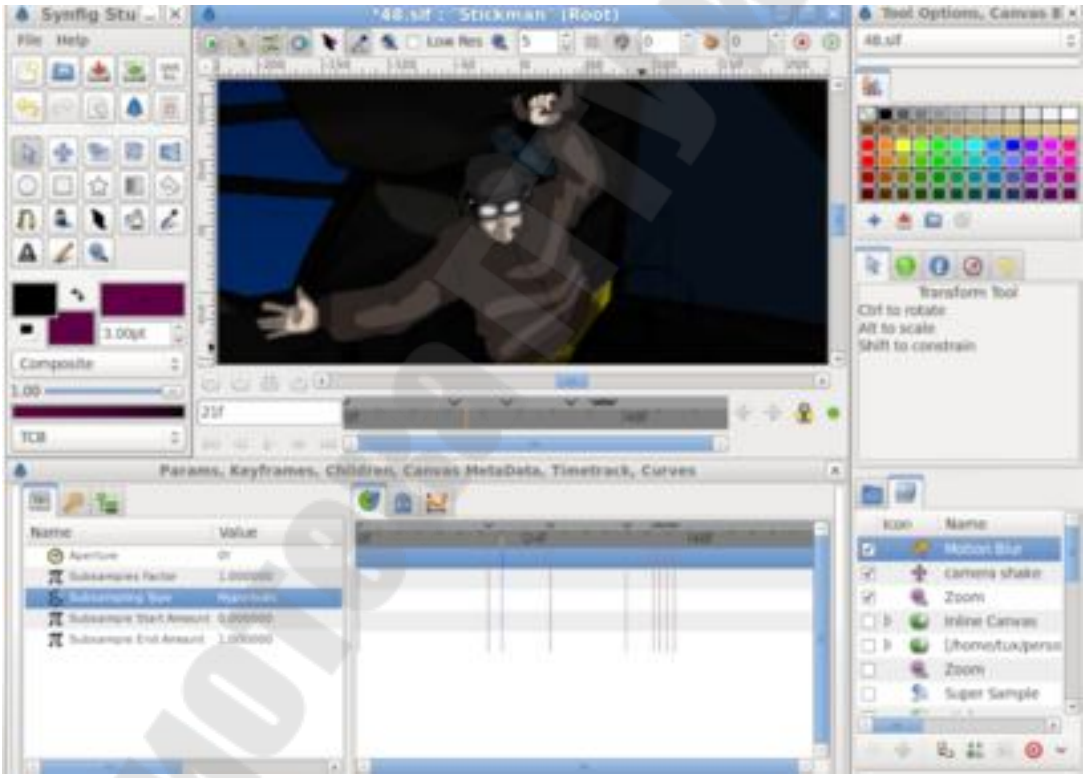


Рисунок 3.3 – Рабочее окно программы Synfig Studio

6) Blender – бесплатная

БЕСПЛАТНЫЕ АНАЛОГИ

SketchUp, Make Art of Illusion, Wings 3D
ПЛАТНЫЕ АНАЛОГИ
3ds Max
ОПЕРАЦИОННЫЕ СИСТЕМЫ
Windows Linux FreeBSD Mac OS X

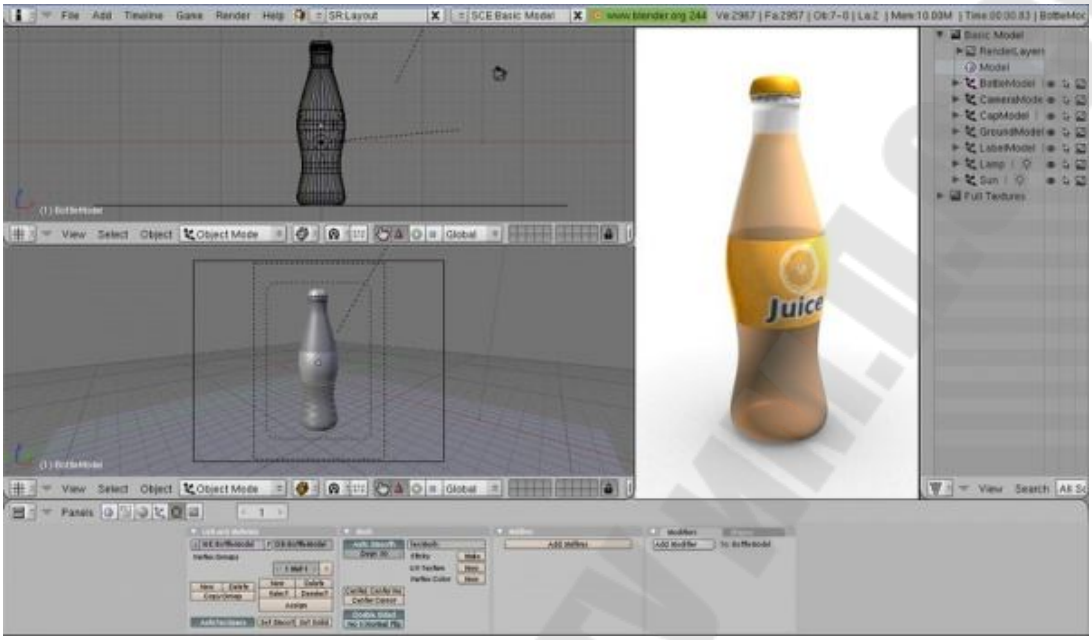


Рисунок 3.4 – Рабочее окно программы Blender

- Поддерживаемые 2D форматы: TGA, JPG, PNG, OpenEXR, DPX, Cineon, Radiance HDR, Iris, SGI Movie, IFF, AVI и Quicktime GIF, TIFF, PSD, MOV (Windows и Mac OS X).
- Поддерживаемые 3D форматы: 3D Studio, AC3D, COLLADA, FBX Export, DXF, Wavefront OBJ, DEC Object File Format, DirectX, Lightwave, MD2, Motion Capture, Nendo, OpenFlight, PLY, Pro Engineer, Radiosity, Raw Triangle, Softimage, STL, TrueSpace, VideoScape, VRML, VRML97, X3D Extensible 3D, xfig экспорт.

- Blender – один из самых многофункциональных бесплатных 3D редакторов. Редактор умеет практически всё что нужно, а именно: создавать модели и анимацию, текстурирование, освещение, предлагает различные материалы. Интерфейс программы довольно стандартный для 3D редакторов, т.е. он имеет большое количество панелей и кнопок. Придётся долго разбираться, чтобы понять, что для чего нужно.

- Стоит отметить, что для разных режимов работы можно выбрать различные интерфейсы: для анимации один, для моделирования другой.
- Для моделирования Blender поддерживает множество стандартных геометрических форм, кривые Безье, NURBS поверхности, скульптурное моделирование, subdivision surface, интерактивное раскрашивание вершин, быстрое создание скелета и многое другое.

Рисунок 3.5 – Рабочее окно программы Blender

В отличие от многих бесплатных 3D редакторов, Blender ещё умеет и создавать анимацию. Для этого программа поддерживает: скелетную анимацию, нелинейную анимацию (и редактор для этого), морфинг, инверсную кинематику, различные привязки ключевых кадров, скриптовый язык Python и многое другое. Типичный экран создания анимации представлен ниже.

Про все функции редактора не рассказать, т.ч. придётся их только перечислить: физика и частицы, 3D в реальном времени и создание игр, рендеринг и шейдинг.

ЗАКЛЮЧЕНИЕ

Blender – очень хороший бесплатный аналог платных 3D редакторов, таких как 3D Max. Возможно, его интерфейс будет непривычен и придётся переучиваться для работы с ним.

Рисунок 3.6 – Рабочее окно программы Blender

7) Pivot Stickfigure Animator – бесплатная

Pivot Stickfigure Animator самая простая и, наверное, самая доступная программа для двумерной анимации (интерфейс упрощен настолько, что любой совершенно неопытный новичок разберется как пользоваться Pivot и что он умеет). Занимает мало места и имеет самые минимальные требования к компьютеру. В первую очередь Pivot подойдет для начинающих художников-аниматоров, а именно для обучения жестовой анимации. Шаг, бег, прыжок, поклон, элементарные действия и жесты. Все это и многое другое можно освоить на уже готовых шаблонах.

Основные герои это человечки, состоящие из палочек и кружочков. У человечков есть локтевые и коленные суставы (суставы на

анимации выглядят красными точками), «шейный сустав», позвоночник неподвижен (количество суставов можно увеличивать по желанию в панели управления персонажем, об этом ниже). Человечки могут быть как черными (по умолчанию) так и других цветов (стандартная палитра имеется). Главным героем может быть как один человечек, так и довольно большая группа.

Теперь по работе с персонажем. В программе Pivot Stickfigure Animator есть несколько типов персонажей, по умолчанию стоит худой человечек. Для того чтобы работать с другим типом персонажа (ковбой, лошадь, слон и так далее) его нужно загрузить из библиотеки которая как правило идет в сборке с программой (путь к ней указывает автоматически при нажатии на нужный пункт меню).

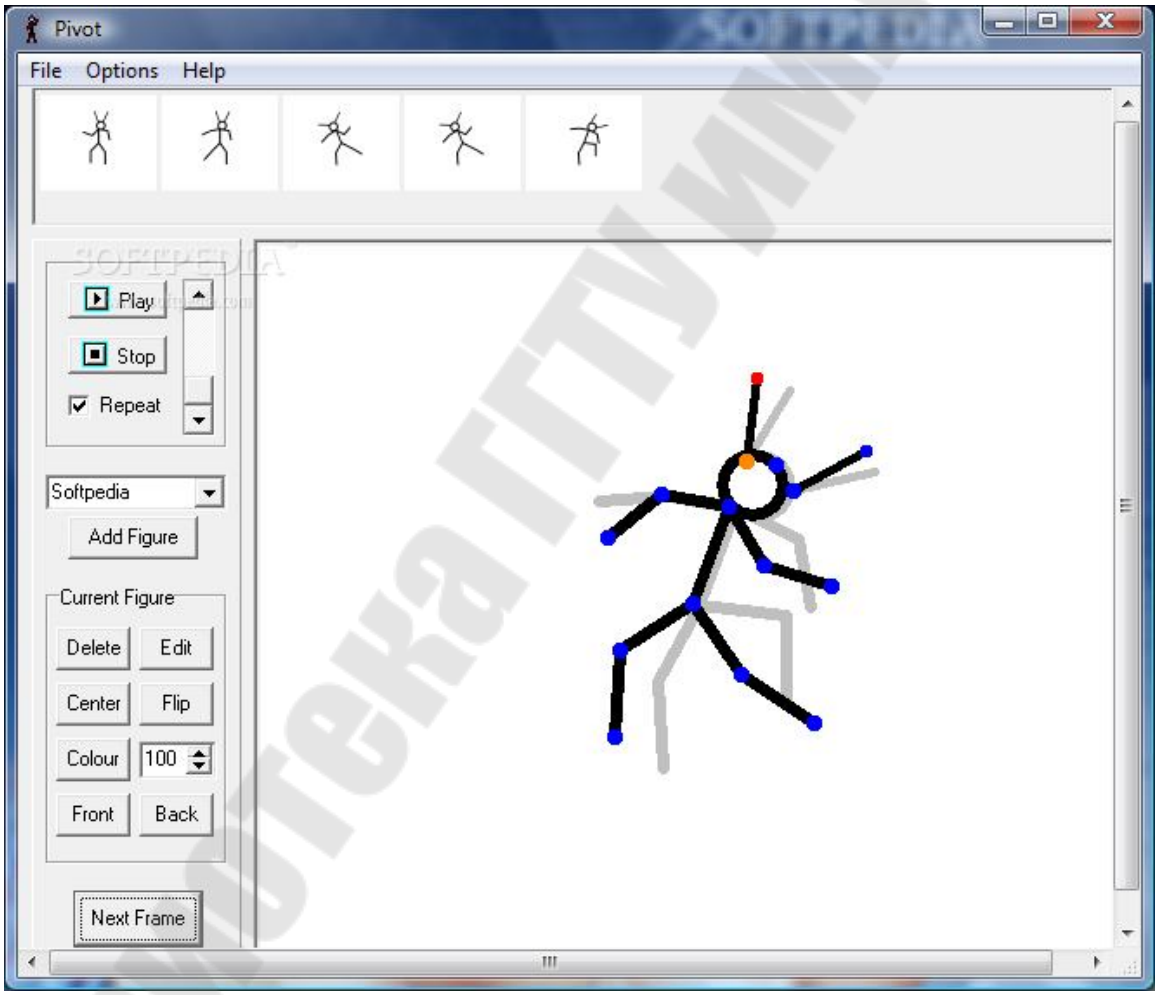


Рисунок 3.7 – Рабочее окно программы Pivot Stickfigure Animator

8) Anime Studio (платн)

программ для создания качественной 2D анимации. Данная программа идеально подходит для профессионалов, ищущих более эффективную альтернативу для создания анимации без утомительной детальной покадровой обработки. С интуитивно понятным интерфейсом и библиотекой готовых персонажей и дополнительных объектов (мультипликационные объекты, фоны, картинки и т.д.) программа предоставляет продвинутые анимационные инструменты и эффекты, которые повысят скорость создания анимации.

Программа объединяет в себе новейшие функции с мощными технологиями для создания уникальных анимаций цифровыми художниками. Для создания анимации вы сможете использовать библиотеку объектов и встроенные инструменты для создания собственных объектов. Программа поддерживает работу со слоями. Готовый проект вы сможете сохранить в качестве видеоролика, графического изображения или SWF-файла.

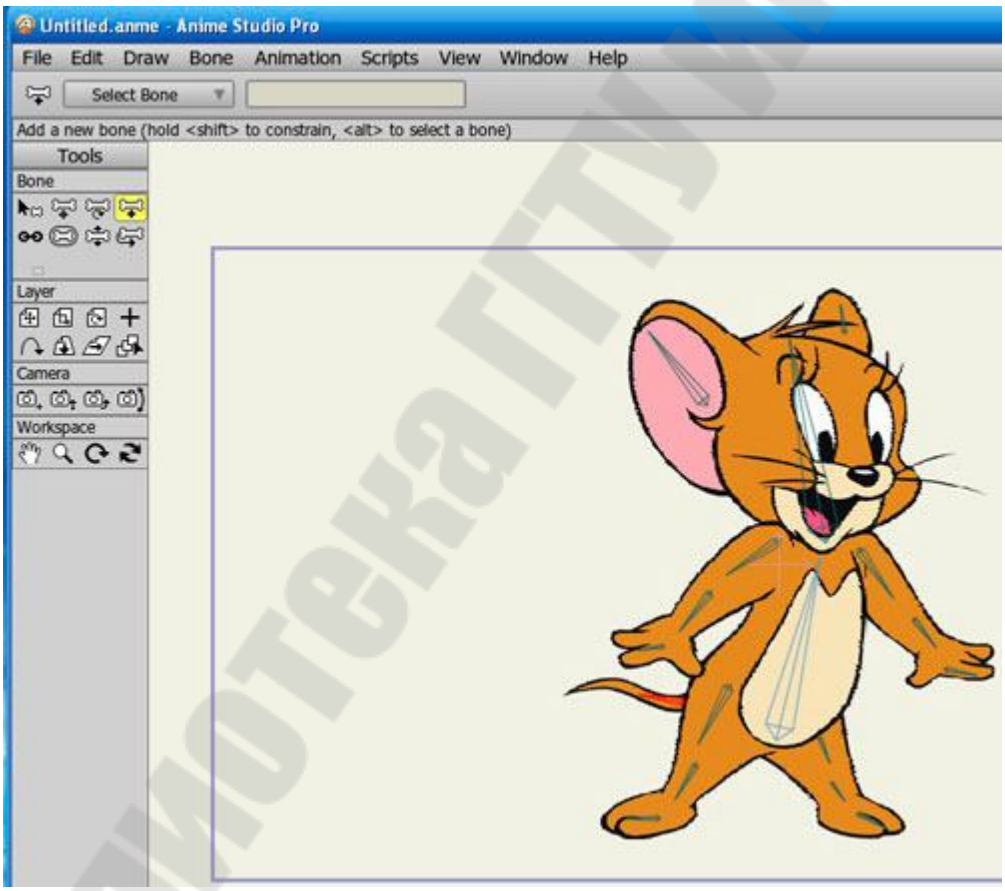


Рисунок 3.8 – Рабочее окно программы Anime Studio

Возможности программы:

– Добавление специальных эффектов к вашим анимациям.

– Сохранение видео и анимации в различные форматы.

– Создание анимаций используя несколько технологий, сохраняющих ваше время.

– Создание скелета с помощью уникальной технологии основанной на построении костей.

– Поддержка создания слоев.

– Анимация отдельных точек на объекте.

– Поддержка документов Adobe Photoshop.

– Скриптинг с помощью LUA 5.1.

– Встроенная библиотека визуальных объектов.

– Полная поддержка импорта и экспорта HD видео файлов.

– Имитация физики: управление взаимодействием объектов, с возможностью настройки плотности, гравитации, трения, и упругости.

– Создание трехмерных объектов на основе двухмерных.

– Инструменты анимации кривых.

– Загрузка файлов в Интернет при помощи Stuffit Connect облегчает совместную работу над проектами.

9) 3ds Max - программа для создания 3D-моделей и визуализации

Самым популярным на сегодняшний день является программный продукт американской компании Autodesk - 3D Studio Max.

Сразу отметим, что идеальной программы по 3D-моделированию, анимации, рендерингу (визуализации) нет и не может быть; каждое приложение имеет свои сильные и слабые стороны, не говоря уже о субъективном восприятии того или иного приложения. Тем не менее, программа 3ds Max имеет ряд объективных преимуществ перед подобными продуктами, скажем, Autodesk Maya. Если последняя программа сильна в основном в анимации, то приложение 3D Studio Max с легкостью справляется и с тем, и с другим, и с третьим, то есть оно практически может все.

3ds Max является «пионером» 3D-моделирования, анимации и рендеринга.

Ре́ндеринг – термин в компьютерной графике, обозначающий процесс получения изображения по модели с помощью компьютерной программы.

Еще одной причиной популярности данного приложения является его огромный инструментарий, а нехватка какого-то специфического, но необходимого Вам инструмента легко компенсируется широчайшей базой плагинов - этих замечательных дополнений, которые могут расширять стандартные возможности Autodesk 3ds Max безгранично много.

Рисунок 3.9 – Выполнено в 3ds Max

В 3D Studio Max легко управлять частицами, что позволяет создавать поразительные эффекты, например, имитацию каких-либо природных явлений: дым от костра, водяные брызги, капельки росы, скатывающиеся с листьев, и так далее. А модуль HairandFur настолько «гениален» в создании волос, что его уже окрестили онлайн-цирюльником: с помощью этого плагина можно не только создавать виртуальные волосы, но также их укладывать, подрезать и делать любую модную прическу.

Майя считается удобнее и сильнее в анимации, поскольку обладает большим числом качественных инструментов для нее. Однако последняя версия Autodesk 3D Studio Max обзавелась прекрасными инструментами скиннинга Heatmap и Geodesic Voxel, контроллерами

анимации в среде создания графов, более тонким инструментом для работы с текстами, отчего ее производительность и возможности в создании анимации повысились.

Зато в моделировании 3д Максу нет равных. Огромный набор инструментов, плагинов позволяет с помощью этого приложения легко моделировать и визуализировать любые интерьеры и самые сложные архитектурные построения. Не случайно 3ds Max Design является любимой программой дизайнеров и архитекторов (она даже интегрирована с AutoCAD), а обозреватели приложений по 3D графике отмечают, что она рассчитана в первую очередь на «технарей». Наверное, поэтому данную программу предпочитают и разработчики всевозможных игр…

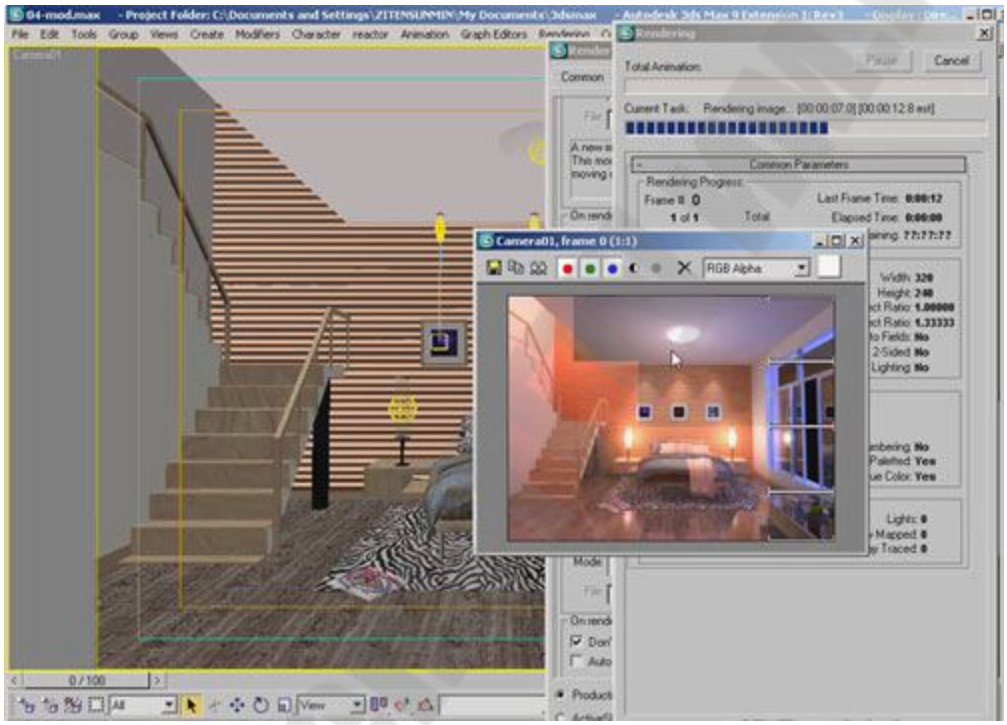


Рисунок 3.10 – Разработка помещений в 3ds Max

10) Autodesk Maya – программа для создания 3D-моделей и визуализации

Одна из важнейших особенностей, присущих программе Autodesk Maya, заключается в том, что данное программное обеспечение является открытым для сторонних разработчиков. Поэтому они могут заниматься преобразованием исходного продукта в специализированную версию, которая наиболее оптимальным образом подхо-

дит для той или иной студии. Таким образом, существует возможность написания специфичного программного кода для определенных нужд.

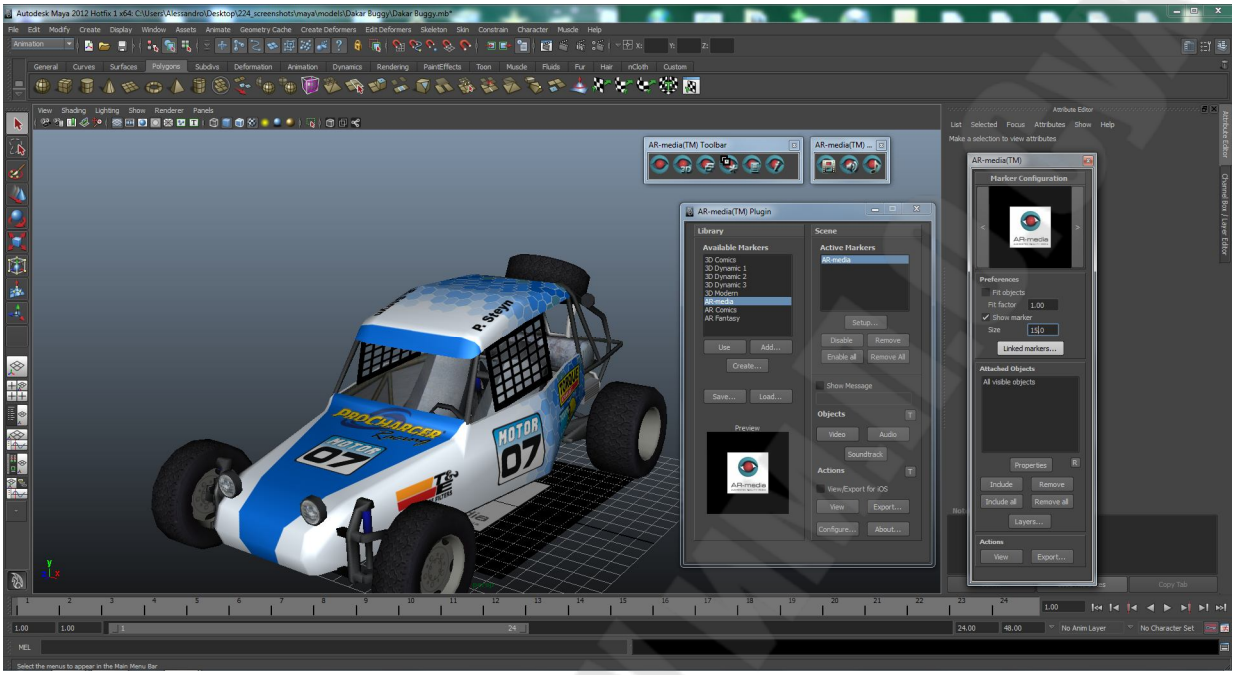


Рисунок 3.11 – Рабочий экран Maya

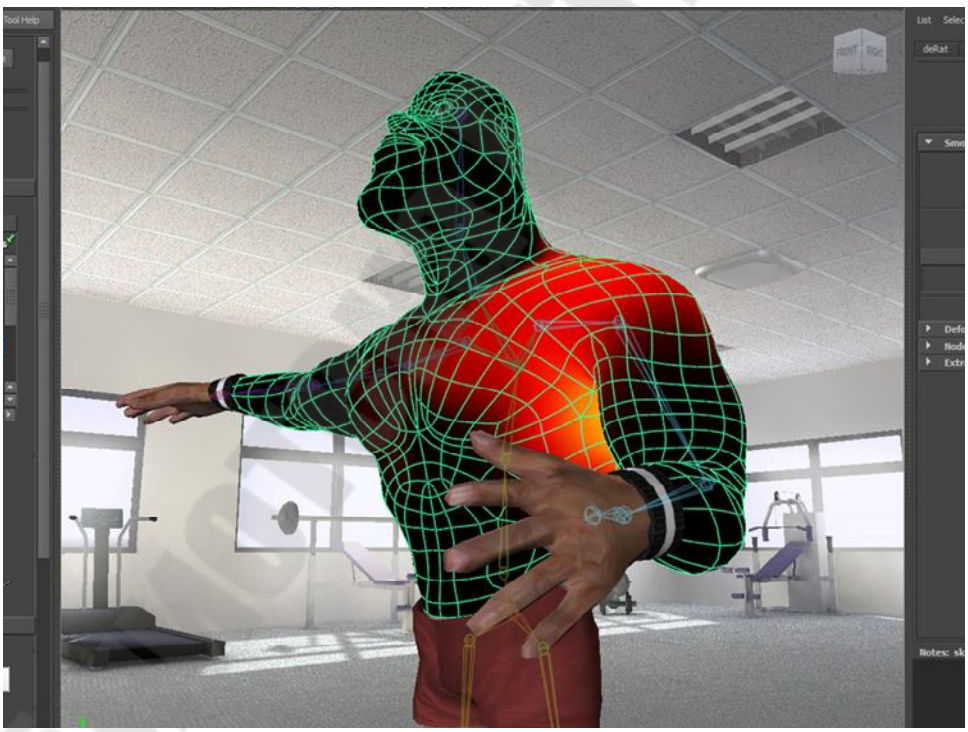


Рисунок 3.12 – Разработка персонажа в Maya

Благодаря встроенному в данную программу интерпретируемому платформенно-независимому языку Maya Embedded Language, являющему достаточно мощным средством для настройки и доработки функционала, пользователь может выполнять запись своих действий в качестве скрипта на языке MEL, что впоследствии можно без проблем превратить в удобный макрос. Поскольку MEL не привязан к определенной платформе, написанные на нем коды легко исполняются в любой из операционной систем, где может полноценно работать утилита Autodesk Maya. В интернете вы можете программу maya скачать и пользоваться бесплатно.

Современное ПО Autodesk Maya представляет собой кроссплатформенное приложение с широкими функциональными способностями, предназначенными для моделирования, композинга, рендера, анимации, а также для многих других задач, связанных с созданием компьютерной трехмерной графики.

Благодаря привлекательному интерфейсу, а также наличию разнообразных инструментов в виде кнопок с узнаваемыми графическими изображениями, пользоваться данной программой очень легко и удобно.

11) Softimage

Softimage – программное обеспечение для создания 3D-анимации для профессионального использования на телевидении, разработки игр, и снятия фильмов. Автономный пакет программного обеспечения для создания анимации с инструментами и приложениями. Оно является лучшим вариантом для аниматоров и моделистов. В состав программного обеспечения входят онлайн-учебники, призванные помочь новичкам.

12) Modo

Modo – профессиональное программное обеспечение для создания 3D-анимации с прекрасными возможностями для лепки, живописи и моделирования. Modo отличается уникальным дизайном с минимальным количеством инструментов для более эффективной работы. *Modo* – полезное приложение для художников и дизайнеров. Оно помогает сделать незаметной разницу между фотографией и компьютерной графикой.

13) Autodesk MotionBuilder

Autodesk MotionBuilder – ведущий программный продукт для 3D-анимации в реальном времени, используемый при разработке компьютерных игр, съемке фильмов и подготовке телепередач. Autodesk MotionBuilder является профессиональным программным средством с инструментами сбора 3D-данных, их обработки и визуализации.

Новая концепция нелинейного редактирования в виртуальных постановках не ограничивается рамками видео и аудио. Autodesk® MotionBuilder®, предназначенный для анимации 3D-персонажей в реальном времени, основывается на перспективных технологиях, которые уже внедрены на ряде ведущих в отрасли студий. Вы можете собирать, объединять и обрабатывать данные с помощью надежных и точных процедур; создавать, редактировать и воспроизводить комплексные анимации персонажей в интерактивной среде с возможностью обратной связи; настраивать отображение в соответствии с индивидуальными потребностями художников-аниматоров и режиссеров-постановщиков. Благодаря всему этому MotionBuilder является идеальным средством для выполняемой в фильмах и играх анимации, предварительной визуализации, захвата движения и анимационных постановок.

14) Houdini

Houdini – профессиональный софт для **3D Artist**, разработанный **Side Effects Software** (SideFx). **Houdini** очень востребованный программный пакет, позволяющий создавать продвинутые эффекты, имеет мощный рендеринг и моделирование. **Houdini** отличается от своих конкурентов тем, что он не нуждается в дополнительных аддонах, все стоит на борту программы. Чаще всего **Houdini** используется для создания эффектов в фильмах, для создания рекламы и игровых объектов.

Houdini — это передовое программное обеспечения для создания спецэффектов. Он может всё: пожары, взрывы, разрушения и симуляции жидкостей. Вы можете использовать инструменты по умолчанию или написать скрип, который поможет вам сделать симуляции.

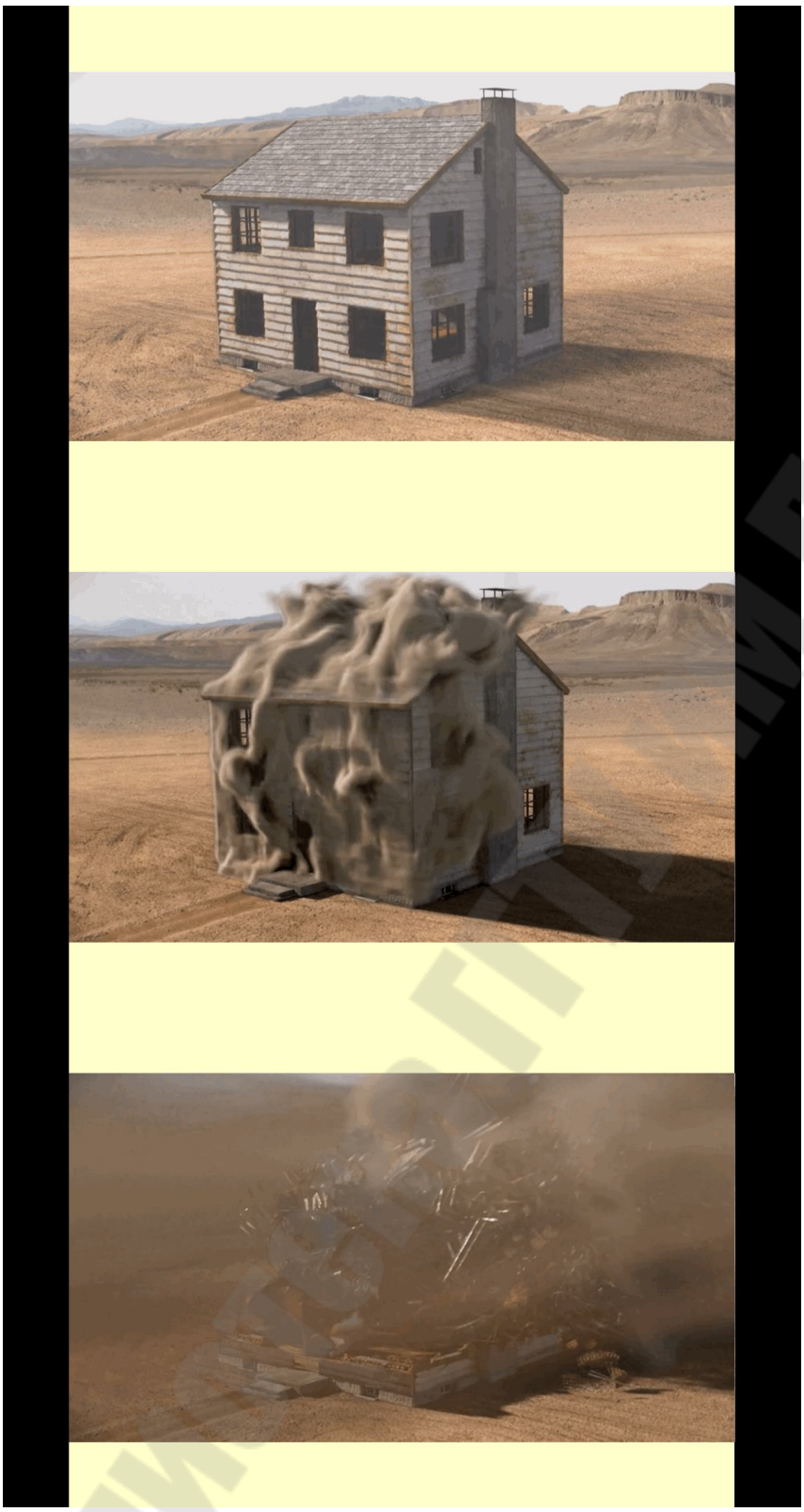


Рисунок 3.13 – Эффекты, созданные в Houdini

15) Cinema 4D studio

У *Cinema 4D studio* приятный и понятный пользовательский интерфейс. Вы можете создать анимированного персонажа со своими характерными движениями. Это программное обеспечение – лучший друг художника-графика. Оно было разработано с нуля с помощью анимационной графики.

Рисунок 3.14 – Рабочий экран Cinema 4D studio

16) Aurora 3D Animation Maker

Aurora 3D Animation Maker – программа, предназначенная для создания красивых трехмерных анимаций, интерактивных логотипов, титров и даже анимированных роликов. Приложение предоставляет пользователю богатый выбор графических эффектов, таких как: салют, дым, пламя, снег и несколько десятков "переходов" между слайдами. Программа также позволяет полностью настроить готовую анимацию, в том числе указать ее продолжительность, степень сжатия, скорость и плавность "воспроизведения".

Кроме того, Aurora 3D Animation Maker предоставляет инструментарий для быстрого размещения готовой анимации на веб-сайте. Сохранить готовый проект можно в форматах AVI, MPG, FLV, SWF (Flash), в качестве анимированного GIF или как изображения в форматах PNG, JPEG, TGA и BMP. К интерфейсу программы претензий нет.

Язык Русский, Английский и другие
Лицензия Бесплатная
Размер 44 Мб (Версия16.03 от 16.02.2017)

17) CrazyTalk

CrazyTalk – приложение, которое позволяет создавать говорящие трехмерные персонажи. С помощью приложения можно загрузить свое лицо и, беря за основу его контуры, создать анимированного персонажа.

Возможности

Программа позволяет создавать анимированные картинки из фотоснимков. Работать в программе можно в двух и трехмерном режиме. Также есть возможность изменять прическу, макияж и аксессуары. Приложение располагает встроенным синтезатором речи. Помимо этого, есть возможность работать со скриптами.

Из отрицательных моментов стоит отметить довольно высокую стоимость программы.

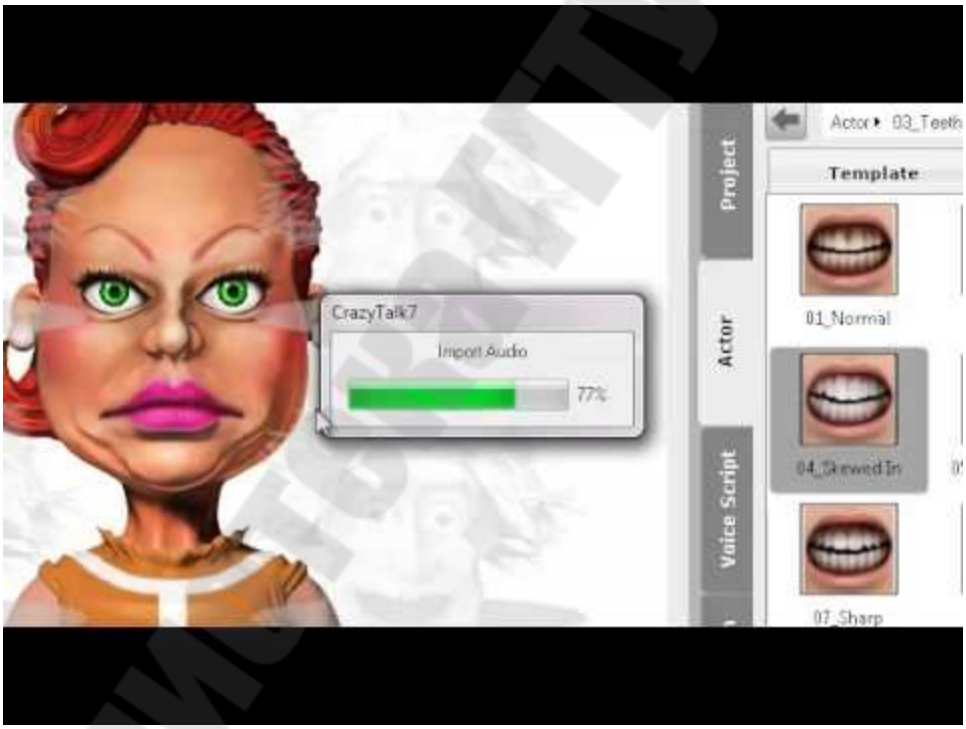


Рисунок 3.15 – Рабочий экран программы CrazyTalk

18) Adobe Premiere Pro

Adobe Premiere Pro – программа, которая позволяет выполнять нелинейный видеомонтаж. Адоб Премьер Про является идеальным инструментом для любого процесса пост-обработки или монтажа отснятого видео.

Очень существенным преимуществом данной программы является её свободная интеграция с другими продуктами Adobe. Большинство профессионалов используют связку Premiere Pro + After Effects + Photoshop. Такой программный пакет решает практически все задачи, с которыми может столкнуться профессиональный работник телевидения или простой любитель видео.

19) Animate CC

Компания Adobe официально выпустила Animate CC – последнюю версию программы для создания веб-анимации. Animate CC ранее была известна как Flash Professional, однако в последние годы технология Flash растеряла свою популярность. Поэтому разработчики решили, что продукт нуждается в переименовании. Animate CC и инструмент для веб-дизайна Muse CC доступны всем подписчикам Creative Cloud.

20) iClone Pro

iClone Pro – приложение для изготовления анимационных роликов с большим количеством профессиональных средств. Они помогают создавать качественные мультфильмы.

С помощью программы можно обрабатывать персонажей, ландшафт, текстуры и другое. В приложении имеется встроенная библиотека реквизитов. Особенностью приложения является возможность преобразования движений людей в анимацию с помощью технологии Kinect от Microsoft.

Приложение предназначено для создания анимационных видеороликов. Утилита обладает пакетом предустановленных элементов. Кроме того, утилита способна распознавать и преобразовывать реальные движения. Помимо этого, приложение способно работать совместно с двухмерными редакторами графики. Также утилита способна присваивать персонажам реальные лица людей.

Рисунок 3.16 – Рабочий экран программы iClone Pro

21) Zbrush

Программа ZBrush от компании Pixologic – это мощный профессиональный инструмент для создания и редактирования трехмерной графики. Для того, чтобы лучше подготовить новичков к работе в приложении, мы подготовили обзор программы ZBrush. В первую очередь программа направлена на работу с так называемой «цифровой глиной», из которой можно буквально вылепливать объекты при помощи разнообразных инструментов. Аналогов такому подходу практически нет в других пакетах для 3D моделирования, хотя некоторые приложения (к примеру, Maya) предлагают специальные инструменты для скульптинга.

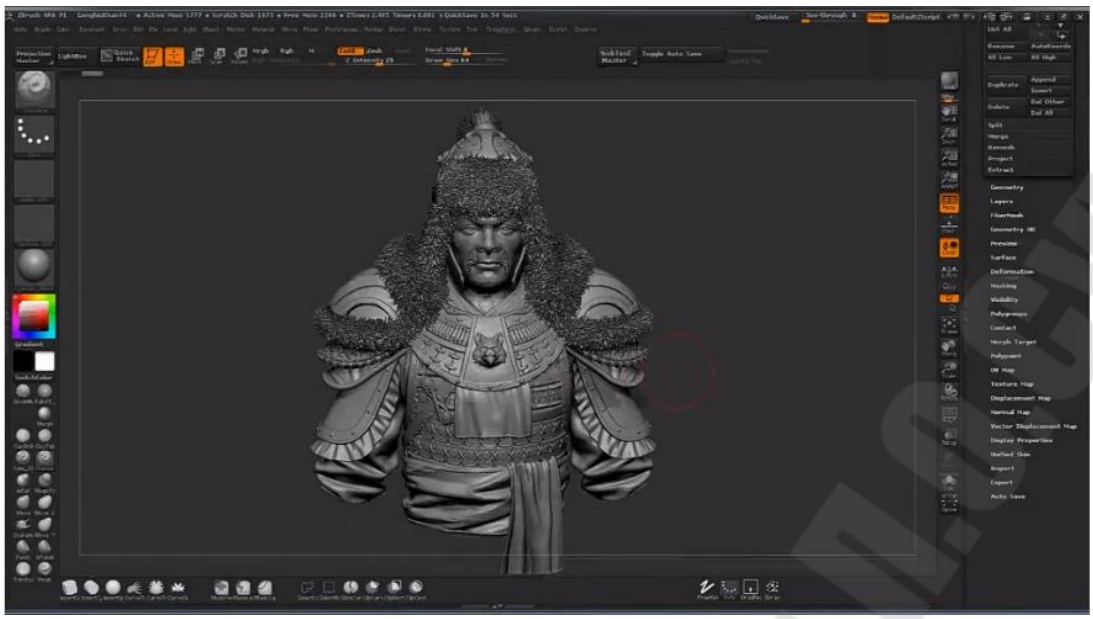


Рисунок 3.17 – Рабочий экран программы ZBrush

Такая цифровая лепка идеальна для создания людей, животных, и вообще всего органического. Тем не менее, ZBrush может использоваться для твердотельного 3D моделирования, и оснащен для этого специальными инструментами. Огромный набор специальных кистей направлен на достижение максимальной реалистичности при создании 3D моделей, а инструменты наложения текстур и визуализации дополняют функционал программы.

Рисунок 3.18 – Рабочий экран программы ZBrush

ZBrush существенно отличается от остальных программ для 3D моделирования, ведь помимо традиционной работы с трехмерной графикой, здесь используется так называемая псевдотрехмерность, или 2,5D. За счет этой уникальной особенности ZBrush практически не использует возможности видеокарты, что существенно влияет на скорость работы.

Как было сказано выше, в ZBrush пользователь работает с цифровой глиной. Весь этот процесс носит название «скульптинг», по понятным причинам. Программа оснащена множеством удобных функций и инструментов для скульптинга, большинство из которых представлены в виде так называемых кистей. С их помощью художник деформирует поверхность модели по своему усмотрению, формируя желаемый объект.

Такой подход оптимален для органики, но также неплохо подходит для твердотельного (Hard Surface) моделирования. Помимо скульптинга программа предлагает возможность обыкновенного 3D моделирования при помощи специального инструмента ZModeler. Он представляет собой особую кисть, способную выполнять разнообразные функции, которые мы привыкли видеть, скажем, в 3DsMax, Maya или Blender.

22) Toon Boom Harmony

Программа для создания анимации, используемая многими профессиональными студиями.

Это многофункциональное приложение для создания качественных мувиклипов и анимационных роликов. Программа содержит множество различных инструментов для разных типов проектов - специальную технологию сканирования объектов с кальки TruePencil, функцию импорта цифровых изображений с графических планшетов, возможность ротоскопинга (покадрового рисования актеров на основании слоев видео), поддержку цифровой аппликации, обработку кукольных и скелетных персонажей и другие опции. С помощью Тун Бум Хармони можно создавать как классические мультфильмы, так и рекламную 3D-анимацию.

Приложение используют студии с мировым именем – Universal, Walt Disney, DreamWorks Animation и другие. Достаточно сказать, что программа применялась при создании культовых мультипликацион-

ных фильмов «Король лев», «Русалочка», а также сериалов «Симпсоны», «Гриффины» и других.

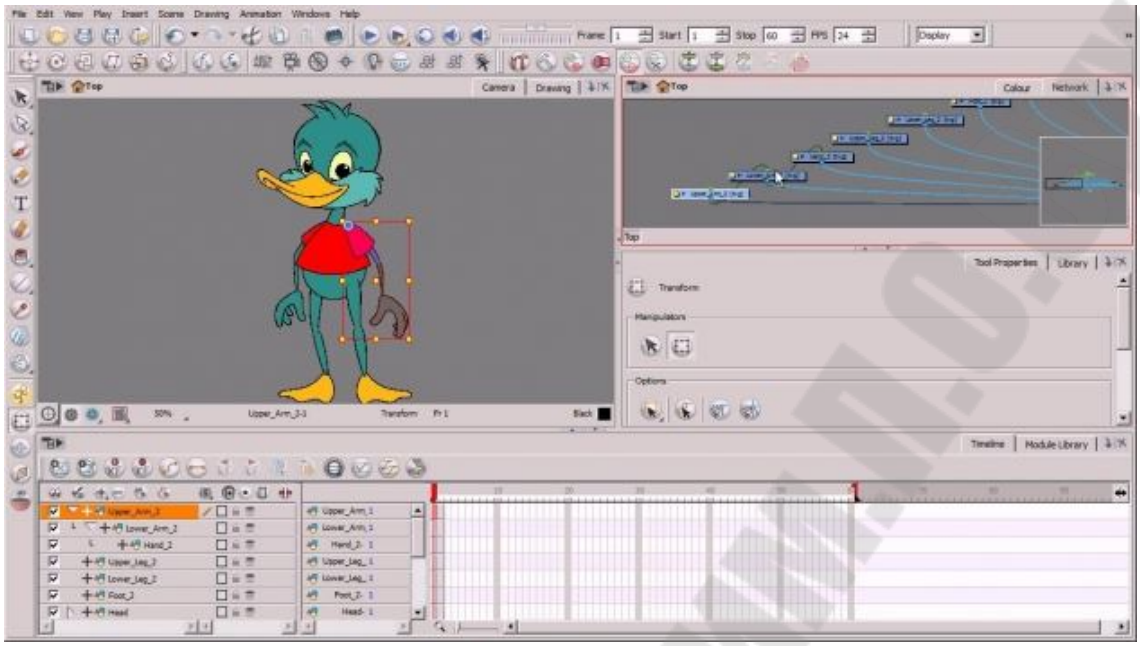


Рисунок 3.19 – Рабочий экран программы Toon Boom Harmony

Коротко и ясно:

- набор инструментов для рисования - от палитры до готовых элементов;
- поддержка разных стилей анимации;
- преобразование объектов на плоскости в 3D-модели;
- оптимизация рисованных «от руки» объектов;
- прямая запись с камеры.

23) Easy GIF Animator Pro

Самой популярной программой для создания GIF изображений, является Easy GIF Animator. Программой просто и легко пользоваться, а главное, разработчиком хорошо получилось совместить простоту с функционалом. Программа поможет сделать из видео файла, анимированное GIF изображение. Вам достаточно открыть видео в программе и выбрать нужный отрезок, дальше программа сама создаст GIF изображение. Также в программе большое количество дополнительных эффектов, которые помогут сделать анимированное изображение лучше. Данную программу можно использовать для создания различных баннеров, всевозможных кнопок и т.п. Если вам нужно

создать движущейся текст, то Easy GIF Animator с этим тоже отлично справляется.

Возможности программы Easy GIF Animator 6

– Создание GIF изображений из фото и видео.
– Создание анимированных баннеров, кнопок и т.п.
– Редактирования уже созданных GIF изображений.
– Большое количество эффектов.
– Создание анимированного текста.
– Высокая степень сжатия для уменьшения размера.
– Оптимизация изображений для быстрой работы.
– Изменения размеров GIF изображений.
– Извлечение кадров.
– И многое другое…

24) Express Animate

Довольно простая программа, которая поможет создать и обработать анимацию и экспортировать результат работы в видеоформат, флеш-анимацию или анимированный GIF.

К своему проекту в Express Animate вы сумеете добавить фоновый звук и сопроводительный текст, а также использовать различные эффекты.

25) Hippani Animator

Hippani Animator – приложение, которое позволяет конструировать анимации и перегонять полученный результат в видео, GIF-файлы или в код для интернет-проводника на языке HTML. Утилита позволяет использовать сложные геометрические фигуры для создания простых роликов.

Утилита способна работать с языком HTML5, также обладает возможностями совместимости работы со структурой CSS. Кроме того, приложение способно работать с интернет-приложениями для мобильных устройств.

Возможности

Программа способна конструировать простые анимационные ролики на базе текста, изображений и геометрических фигур. Кроме того, есть возможность преобразовывать проекты в HTML-код, видеоролики и другое. Помимо этого, имеется встроенный редактор

картинок. Также утилита способна работать с элементами на JavaScript. К тому же есть возможность использовать аудио, видео и другое.

Рисунок 3.20 – Рабочий экран программы Hippani Animator

Плюсы и минусы

Приложение способно работать с векторной графикой. Кроме того, все команды логично обозначаются. Помимо этого, программа может работать со всеми устаревшими браузерами. Используя это приложение, можно создавать сложные заставки для мобильных устройств.

Из отрицательных моментов стоит выделить тот факт, что для создания анимаций необходимо иметь некоторый опыт. Кроме того, в программе нет русскоязычной локализации.

Лекция 4. Основы CSS-анимации

Наверное, следует согласиться с тем, что CSS-анимация немного ограничена по сравнению с js-скриптами. Но в тоже время анимация с использованием js-скриптов больше нагружает аппаратную часть. CSS-анимацией сложно управлять. Анимацией, выполненной при помощи скриптов, управлять гораздо удобнее. В конце концов, выбор средства остается за разработчиком. Мы отметим лишь, что современный CSS3 предлагает очень широкие возможности для добавления анимации в веб-проект. В этой и следующей главе мы подробно будем подробно знакомиться с основными приемами CSS-анимации.

Часть 1. CSS – переходы

CSS3-переходы позволяют анимировать исходное значение CSS-свойства на новое значение с течением времени, управляя скоростью смены значений свойств. Большинство свойств меняют свои значения за 16 миллисекунд, поэтому рекомендуемое время стандартного перехода — 200ms.

Смена свойств происходит при наступлении определенного события, которое описывается соответствующим псевдоклассом. Чаще всего используется псевдокласс :hover. Данный псевдокласс не работает на мобильных устройствах, таких как iPhone или Android. Универсальным решением, работающим в настольных и мобильных браузерах, будет обработка событий с помощью JavaScript (например, переключение классов при клике).

Переходы применяются ко всем элементам, а также к псевдоэлементам :before и :after. Для задания всех свойств перехода обычно используют краткую запись свойства transition.

CSS3-переходы могут применяться не ко всем свойствам и их значениям.

Переход является простой анимацией от одного набора CSS-свойств к другому через определенный промежуток времени.

Чтобы переход заработал, нужно следующее:

1) два стиля – начальный вид элемента и конечный вид;

2) свойство *transition.* Обычно применяется к исходному стилю;

3) инициатор – представляет собой действие, вызывающее изменение от одного стиля к другому. В CSS для запуска анимации можно использовать несколько псевдоклассов.

Можно анимировать:

- color, background-color;
- border-color, border-width;
- font-size, height, width;
- margin, padding;
- opacity;
- свойства позиционирования.

В основе CSS-переходов лежат четыре свойства, которые управляют тем,

какие свойства анимировать *transition-property;*

продолжительность анимации *transition-duration;*

тип анимации *transition-timing-function;*

задержка перед началом анимации *transition-delay.*

Синтаксис

Элемент{

свойство1: значение;

свойство2: значение;

свойство3: значение;

свойство4: значение;

transition-property: свойство2, свойство4;

}

Элемент:действие{

свойство2: новоеЗначениеСвойства2;

свойство4: новоеЗначениеСвойства4;

}

Анимация цвета – простой переход

```
.b1{
background-color: white;
color: darkblue;
transition-property: background-color, color;
}
.b1:hover{
background-color: darkblue;
color: white;
}
```

В рассмотренном выше примере анимируется элемент, к которому прикреплен CSS-класс .b1 (кнопка, в данном случае). Отметим, что анимируются два свойства: цвет фона и цвет текста. При наведении указателя на кнопку, цвет фона меняется на цвет текста, а цвет текста, в свою очередь, на цвет фона. Анимация происходит сразу же при наведении.

Анимация цвета – добавлено время анимации

```
.b2{
        background-color: white;
        color: darkblue;
        transition-property: background-color, color;
        transition-duration: 3s;
}
.b2:hover{
```

background-color: darkblue;

color: white;

}

Рисунок 4.1 – Анимация цвета

Длительность анимации задается свойством *transition-duration.*

Теперь анимация происходит не мгновенно, а длится в течение трех секунд.

В примере, предложенном ниже, анимируются несколько свойств.

Анимирование трех свойств: color, background-color, opacity

.b3{

background-color: white;

color: darkblue;

opacity: 1;

transition-property: background-color, color, opacity;

transition-duration: 3s;

}

.b3:hover{

background-color: darkblue;

color: white;

opacity: 0;

}

Этот пример отличается от предыдущего тем, что, по происшествии трех секунд, свойство *opacity* кнопки становится равным нулю, а значит, кнопка становится невидимой.

Управление скоростью анимации

Скорость выполнения анимации контролируется с помощью свойства transition-timing-function, которое может принимать значения:

- linear;
- ease (по умолчанию);
- ease-in (медленнее в начале анимации);
- ease-out (медленнее в конце анимации);
- ease-in-out;
- cubic-bezier.

В предложенном ниже примере имеются четыре объекта – четыре рыбки. При наведении курсора мыши на объект, он движется вправо. Какое свойство следует анимировать, чтобы рыбка двигалась вправо? Здесь возможно не одно решение.

```
#fish1{
        position: relative;
        left: 0px;
        transition-property: left;
        transition-duration: 3s;
}

#fish1:hover{
        left: 600px;
}
```

Как видно из этого примера, мы анимировали CSS-свойство *left.*

```
#fish4{
        position: relative;
```

```
        left: 0px;
        transition-property: left;
        transition-duration: 3s;
        transition-timing-function: cubic-bezier(0.8, 0.1, 0.9, 0.1);
    }
    #fish4:hover{
        left: 1000px;}
```

А в этом примере использована еще одна функция скорости – *cubic-bezier*, дающая больший контроль над скоростью анимации.

Иногда необходимо, что бы элемент сохранил тот вид, который был у него по завершении анимации (например, мы хотим, чтобы рыбка осталась справа). Следующая опция заставит браузер сделать это.

```
animation-fill-mode: forwards
```

Наконец, ниже представлен еще один вариант того, как можно заставить объект двигаться вправо. На этот раз анимируется CSS-свойство *margin-left.*

```
    #fish44{
        position: relative;
        /*margin-left: 10px;*/
        transition-property: margin-left;
        transition-duration: 3s;
    }
#fish44:hover{margin-left: 1000px;}
```

Часть 2. CSS – transform

CSS3-трансформации изменяют размер, форму и положение элемента на веб-странице с помощью свойства transform. Трансформации преобразовывают элемент, не затрагивая остальные элементы веб-страницы, т.е. другие элементы не сдвигаются относительно него. По умолчанию трансформация происходит относительно центра элемента.

К элементам, которые могут быть трансформированы, относятся элементы с display: block; и display: inline-block;, а также элементы, значение свойства display которых вычисляется как table-row, table-row-group, table-header-group, table-footer-group, table-cell или table-caption. Трансформированным считается элемент с любым установленным значением свойства transform, отличным от none.

Существуют два вида CSS3-трансформаций – **2D** и **3D**. **2D-трансформации** преобразовывают элементы в двухмерном пространстве.

Допустимые значения:

matrix() – любое число

translate(), translateX(), translateY() – единицы длины (положительные и отрицательные), %

scale(), scaleX(), scaleY() – любое число

rotate() – угол (deg, grad, rad или turn)

skew(), skewX(), skewY() – угол (deg, grad, rad)

Поворот (rotate)

transform: rotate(25deg); поворот на 25 градусов по часовой стрелке;

transform: rotate(-75deg); поворот на 75 градусов против часовой стрелки;

Масштабирование (scale)

transform: scale(2) – увеличение по x и по y в два раза;

transform: scale(2, 0.5) – увеличение в 2 раза по *x* и уменьшение в 2 раза по *y*;

transform: scaleX(3) – увеличение в 3 раза по *x*;

Рисунок 4.2 – Масштабирование элементов веб-страницы

transform: scale(-1) – поворот вокруг обеих осей;

transform: scale(1, -1) – поворот вокруг поворот вокруг оси *x*;

transform: scale(-1) – поворот вокруг обеих осей;

#twit3:active {transform: scale(-1, 1);} – CSS-код для элемента twit3 (на скриншоте видно, что второй элемент, при применении данного кода зеркально отразился справа налево; поворот осуществлен вокруг оси y).

Перемещение (translate)

transform: translate (20px, -10px) – перемещение вправо на 20 px и вверх на 10 px;

transform: translate (2px, 2px) – перемещение вправо на 2 px и вниз на 2 px;

transform: translateY (-30px) – перемещение вверх на 30 px;

transform: translateX (0.5em) – перемещение вправо на половину высоты шрифта;

```
#but:active {
        transform: translate(1px, 2px);
}
```

Приведенный выше CSS-код создает эффект нажатия на кнопку – при этом она смещается на немного вправо и вниз.

Наклон (skew)

#tube1:active {

transform: skew(-45deg, 0);

}

Приведенный выше код наклоняет элемент #tube1 на 45 градусов по часовой стрелке. При этом наклон осуществляется вдоль горизонтальной оси (оси x).

transform: skew (45geg, 0) – на 45 против часовой стрелки;

transform: skew (0, -45deg) – на 45 против часовой стрелки, но на этот раз наклон происходит вдоль вертикальной оси (оси y);

transform: skew (-30geg, -50deg) – наклон по двум осям;

Приведенные выше виды трансформации можно применять одновременно.

Комбинированный эффект

Эффекты перечисляются после *transform* в произвольном порядке.

#witch4:active {

transform: scale(1.8) skew(-15deg, 0) rotate(45deg) translate(-150px);

}

В приведенном выше примере CSS-кода элемент с id witch4 последовательно подвергается следующим изменениям: увеличивается в 1.8 раз, наклоняется вправо на 15 градусов, поворачивается вправо на 45 градусов и, наконец, смещается влево и вверх на 45px.

Свойство transform-origin

Это свойство позволяет изменить точку трансформации объекта, т.е. точку, относительно которой преобразование будет осуществ-

ляться. По умолчанию, эта точка находится в центре элемента. До сих пор вся наша анимация осуществлялась относительно такой точки.

```
#tree2:active {
    transform-origin: 0% 100%;
    transform: rotate(45deg);
}
```

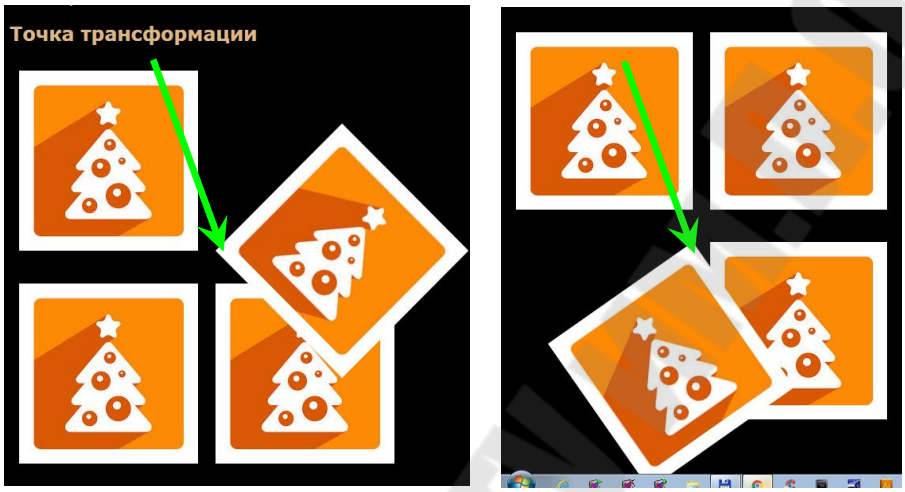


Рисунок 4.3 – Перенос точки трансформации объекта

Объясним этот код. Прежде, чем поворачивать вторую елочку на 45 градусов по часовой стрелке, мы сместили точку трансформации: она перешла в левый нижний угол рисунка (по x – 0%, т.е. перешла в левый край объекта, а по y – 100%, т.е. перешла в самый низ объекта).

```
#tree3:active {
    transform-origin: 100% 0%;
    transform: rotate(-35deg);
}
```

Точка трансформации объекта *#tree3* перешла в правый верхний угол.

Точка трансформации, разумеется, может находиться вообще за пределами элемента.

```
#tree4:active {
```

```
    transform-origin: -400px -400px;
    transform: rotate(-25deg);
}
```

Часть 3. CSS3 – анимация

CSS3-анимация придаёт сайтам динамичность. Она оживляет веб-страницы, улучшая взаимодействие с пользователем. В отличие от CSS3-переходов, создание анимации базируется на ключевых кадрах, которые позволяют автоматически воспроизводить и повторять эффекты на протяжении заданного времени, а также останавливать анимацию внутри цикла.

CSS3-анимация может применяться практически для всех html-элементов, а также для псевдоэлементов :before и :after. Список анимируемых свойств приведен на этой странице. При создании анимации не стоит забывать о возможных проблемах с производительностью, так как на изменение некоторых свойств требуется много ресурсов.

С помощью CSS-переходов можно анимировать *только переход от одного набора CSS-свойств к другому*.

Анимация, предусмотренная в CSS3, позволяет анимировать переходы от *одного набора свойств к другому, затем к третьему…*

Можно задать повторяющуюся анимацию, приостановить ее выполнение при проходе указателя мыши над элементом и даже повернуть ее вспять, когда анимация дойдет до конца.

Анимация с помощью ключевых кадров в CSS3, сложнее перехода, но для нее *не требуется специальный инициализатор* (например, ***hover***).

Создание анимации с помощью *keyframes*

При создании анимации первым делом создаются ключевые кадры *(keyframes).*

Создание анимации происходит в два приема:

1) Определение анимации

```
@keyframes имяАнимации{
    from{список CSS-свойств;}
    to{список CSS-свойств;}
}
```

2) Применение анимации к элементу

```
<div id = “element”></div>
#element:hover {
    animation-name: имяАнимации;
    animation-duration: 4s;
}
```

Рассмотрим несколько примеров.

Анимация цвета

```
@keyframes backgroundGlow{
    from{background-color: black;}
    to{background-color: yellow;}
}
body{
    animation-name: backgroundGlow;
    animation-duration: 1s;
}
```

В первом блоке приведенного выше кода определяется анимация с именем *backgroundGlow*. Анимация призвана изменять цвет фона с черного на желтый.

Во втором блоке эта анимация назначается элементу *body*. Анимация произойдет при загрузке страницы и будет длиться одну секунду.

Анимация цвета с промежуточными кадрами

Можно создавать более двух ключевых кадров, что гораздо более интересно. По такому принципу можно добавлять сколько угодно ключевых кадров – просто используйте необходимое число в процентах (30%, 60%, 75% и так далее).

```
@keyframes color1{
    from{background-color: red;}
    20%{background-color: black;}
    40%{background-color: red;}
    60%{background-color: black;}
    80%{background-color: red;}
    to{background-color: black;}
}
```

Эта анимация заставит цвет фона элемента, к которому она применена, изменяться несколько раз с красного на черный, что создает эффект мигания.

Отметим, что ключевые слова **from** и **to** равнозначны процентным записям **0%** и **100%** соответственно.

Анимация цвета с промежуточными кадрами. Анимация нескольких свойств

```
@keyframes color2{
from{background-color: red;}
50%{background-color: white;
    transform-origin: 0% 100%;
        transform: scale(1.5)}
to{background-color: black;
    transform-origin: 0% 100%;
        transform: scale(2.2)}
```

```
}

@keyframes color3{
from{background-color: red;}
25%{background-color: white;
        transform: scale(1.25);
        transform: rotate(720deg)}
75%{background-color: white;
        transform: scale(1.25);
        transform: rotate(720deg)}
to{background-color: black;
        transform: translate(-200px, -200px)}
}
```

Эти три вида анимации применяются к ячейкам таблицы 4x4.

```
#cell1:hover {
animation-name: color1;
animation-duration: 2s;}

#cell2:hover {
animation-name: color1;
animation-duration: 0.5s;}

#cell3:hover {
animation-name: color2;
animation-duration: 2s;}
```

Рисунок 4.4 – Масштабирование
с помощью ключевых кадров

#cell4:hover {

animation-name: color3; animation-duration: 4s;}

Применение нескольких анимаций

Определим две анимации:

@keyframes smile1{

from{;}

to{transform: rotate(1080deg);}

}

@keyframes smile4{

from{opacity: 1;}

to{opacity: 0;}

}

Следующий код применяет эти анимации к одному объекту:

#s3:hover {

animation-name: smile1, smile4;

animation-duration: 2s, 2s;

}

Управление скоростью анимации

Свойство *animation-duration* позволяет указать продолжительность анимации.

Можно также указывать функцию распределения скорости анимации во времени *animation-timing-function***,** которая может принимать значения

- linear

- ease (по умолчанию)
- ease-in (медленнее в начале анимации)
- ease-out (медленнее в конце анимации)
- ease-in-out

Ниже мы приводим пример, в котором используются эти функции, а также функция cubic-bezier.

```
@keyframes smile3{
from{;}
to{transform: translateX(400px);} }
#cell_a:hover {
animation-name: smile3;
animation-duration: 1.5s;
animation-timing-function: ease-in;}
#cell_b:hover {
animation-name: smile3;
animation-duration: 1.5s;
animation-timing-function: ease-in-out;}
#cell_c:hover {
animation-name: smile3;
animation-duration: 1.5s;
animation-timing-function: linear;}
#cell_d:hover {
animation-name: smile3;
animation-duration: 1.5s;
animation-timing-function: cubic-bezier(1, 0.1, 1, 0.1);}
```

Повторение анимации

С помощью CSS-переходов можно контролировать еще несколько аспектов анимации, включая повторение и направление выполнения.

animation-iteration-count: 5; – повторить анимацию 5 раз;

animation-iteration-count: infinite; – повторить анимацию все время;

animation-delay: 2s; – задержка перед началом выполнения анимации 2 секунды;

animation-direction: alternate; – проиграть анимацию с конца, т.е. анимация проиграется из A в B (как было бы и без этой опции), а затем из B в A.

animation-direction: reverse; – анимация будет проигрываться сразу с конца в начало, т.е. из B в A.

Пример. Пульсирующие кнопки

Создаем две анимации, которые будучи примененными, изменяют масштаб объекта.

```
@keyframes pulse{
from{;}
20%, 40%{transform: scale(1.2);}
25%, 45%{transform: scale(1.0);}
to{;}
}
@keyframes pulse2{
from{;}
10%, 30%, 50%, 70%, 90%{transform: scale(1.05);}
20%, 40%, 60%, 80%, 100%{transform: scale(1.0);}
```

to{;}

}

Вот примеры использования этих видов анимаций. Создан эффект бьющегося сердца.

#hart1:hover {

animation-name: pulse;

animation-duration: 1.5s;

animation-delay: 2s;}

#hart2:hover {

animation-name: pulse;

animation-duration: 1s;

animation-delay: 1s;

animation-iteration-count: infinite;}

#butt:hover {

animation-name: pulse2;

animation-duration: 0.5s;

animation-iteration-count: 10;}

#but:hover {

animation-name: smile4;

animation-duration: 0.5s;

animation-iteration-count: 8;

animation-direction: alternate;}

Приостановка анимации

В приведенном ниже примере, елочка вращается бесконечно, но вращение останавливается при наведении мыши на нее.

```
@keyframes rot{
from{opacity: 1;}
to{transform: rotate(360deg);
        opacity: 0;} }
#tree {
animation-name: rot;
animation-duration: 0.7s;
animation-iteration-count: infinite;
animation-timing-function: linear;}

#tree:hover {
animation-play-state: paused;}
```

Часть 4. CSS3-3d – трансформации

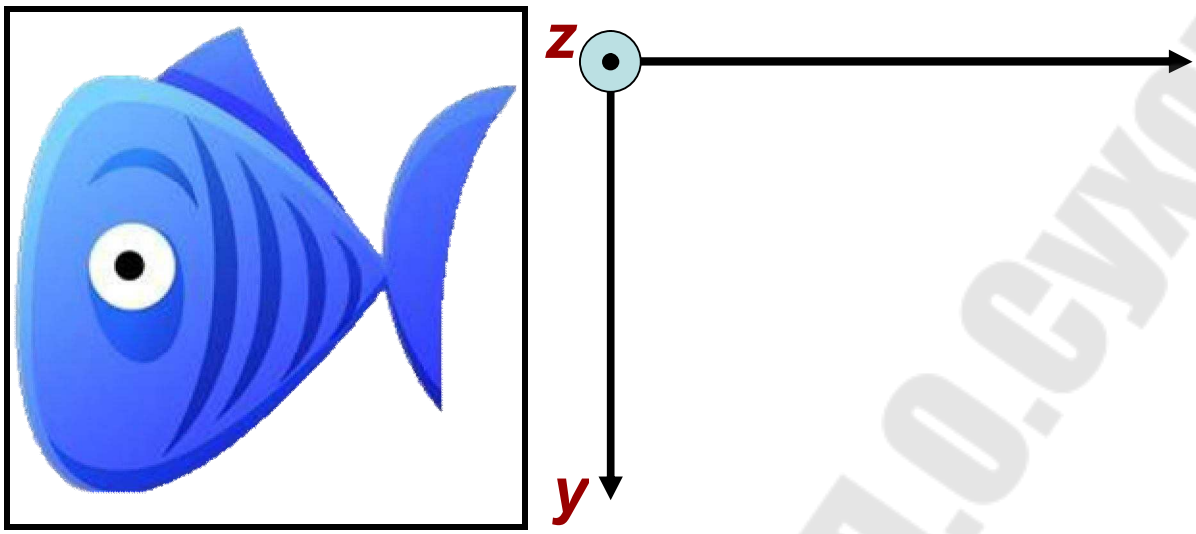
CSS3 3D-трансформации создают объемные реалистичные эффекты на веб-страницах.

3D-трансформации работают аналогично с 2D-трансформациями с разницей в том, что элементы могут перемещаться вдоль оси Z, вглубь экрана и из него.

Чтобы активизировать 3D-пространство, нужно установить свойство perspective для родительского контейнера.

Трехмерные преобразования выполняются в воображаемом трехмерном пространстве, проецируемом на плоскость страницы.

Двумерные преобразования выполняются по горизонтальной и вертикальной координатным осям (*x* и *y*). В случае трехмерных преобразований к ним добавляется ось *z,* направленная из начала координат перпендикулярно воображаемой поверхности элемента.




```
@keyframes fish1{
        from{;}
        to {transform: rotateX (360deg);} }

@keyframes fish2{
        from{;}
        to {transform: rotateY (360deg);} }

@keyframes fish3{
        from{;}
        to {transform: rotateZ (360deg);} }

@keyframes fish4{
        from{;}
        to {transform: rotate3d(1, 1, 1, 360deg)}}
```

Синтаксис

transform: rotate3d(x, y, z, deg);

Первые три параметра определяют, вокруг какой из осей координат будет вращаться объект, а последний – на сколько градусов. X, Y и Z задаются не как абсолютные величины, а как соотношение углов. Например, код transform: rotate3d(2, 1, 0, 90deg); заставит объект повернуться на 90 градусов вокруг оси X и на 45 (90 * 1 / 2) градусов вокруг Y. То же самое сделает и строчка transform: rotate3d(90, 45, 0, 90deg).

Пример

```
@keyframes hell1{
        from{;}
        to{transform: rotate3d(1, 0, 0, 180deg)}
}

@keyframes hell2{
        from{;}
        to{transform: rotate3d(2, 1, 1, 180deg)}
}
```

Перспектива

Перспектива – это расстояние, отсчитываемое по оси z, между воображаемым наблюдателем и поверхностью контейнера, в котором находится подвергаемый преобразованиям элемент. Фактически перспектива добавляет изображению глубину

По умолчанию перспектива равна нулю, т.е. фактически отсутствует. Из-за этого мы получим не тот результат, на который рассчитываем.

Чтобы активизировать 3D-пространство, нужно установить свойство perspective для родительского контейнера.

Свойство активирует 3D-пространство для элемента. Свойство perspective добавляет глубину элементу, увеличивая его размеры по оси Z, сам элемент при этом становится визуально меньше. Чем меньше значение, тем ближе Z-пространство к зрителю и тем больше эффект, заданный с помощью свойства transform.

Свойство perspective активирует 3D-пространство внутри элемента, содержащего дочерние трансформированные элементы и применяется к ним. Свойство не наследуется.

Perspective

Значения:

длина – Задает глубину просмотра, т.е. расстояние по оси Z. Значение может быть любым положительным числом. Если единица измерения не указана, по умолчанию она считается в px. **Чем больше значение, тем менее выражен эффект**. 0 означает отсутствие пер-

спективы. none – Значение по умолчанию. Означает отсутствие перспективы.

initial – Устанавливает значение свойства в значение по умолчанию

Синтаксис

```
ul {perspective: 500px;}
li {transform: rotateX(50deg);}

li:hover {transform: perspective(900px) rotate3d(180,-45,0,-135deg);}
```

Пример

```
@keyframes big1 {
    0% {transform: translate3d(0, 0, 0);}
    100% {transform: translate3d(0, 0, -200px);}
}

@keyframes big2 {
    0% {
    transform: translate3d(0, 0, 0);
    }
    100% {
      transform: translate3d(0, 0, 200px);
}}

div {
perspective: 500px;
}
/*задана перспектива для контейнера div. Теперь две картинки будут перемещаться. Одна к нам, а другая – вглубь*/

#access1 {
  animation: big1 2s infinite alternate;
}

#access2 {
  animation: big2 2s infinite alternate;
}
```

Рисунок 4.5 – Перемещение картинок вдоль оси Z

Задание точки трансформации для 3D-элемента perspective-origin

Свойство устанавливает точку начала координат для свойства perspective. Значение по умолчанию perspective-origin: 50% 50%;. Позволяет изменять направление трансформации дочернего 3D-элемента. Свойство должно использоваться вместе со свойством perspective для блока-контейнера и свойством transform для дочернего элемента.

Синтаксис

ul { perspective: 150px; perspective-origin: 10% 10%; }
li {transform: rotateX(50deg);}

Пример

```
#ul1 {
      list-style-type: none;
      perspective: 200px;
      perspective-origin: 10% 10%}
#ul1 li:hover {
      transform: rotateY(50deg);}
#ul2 {
```

```
        list-style-type: none;
        perspective: 200px;
        perspective-origin: 90% 90%}
#ul2 li:hover {
        transform: rotateY(50deg);}
```

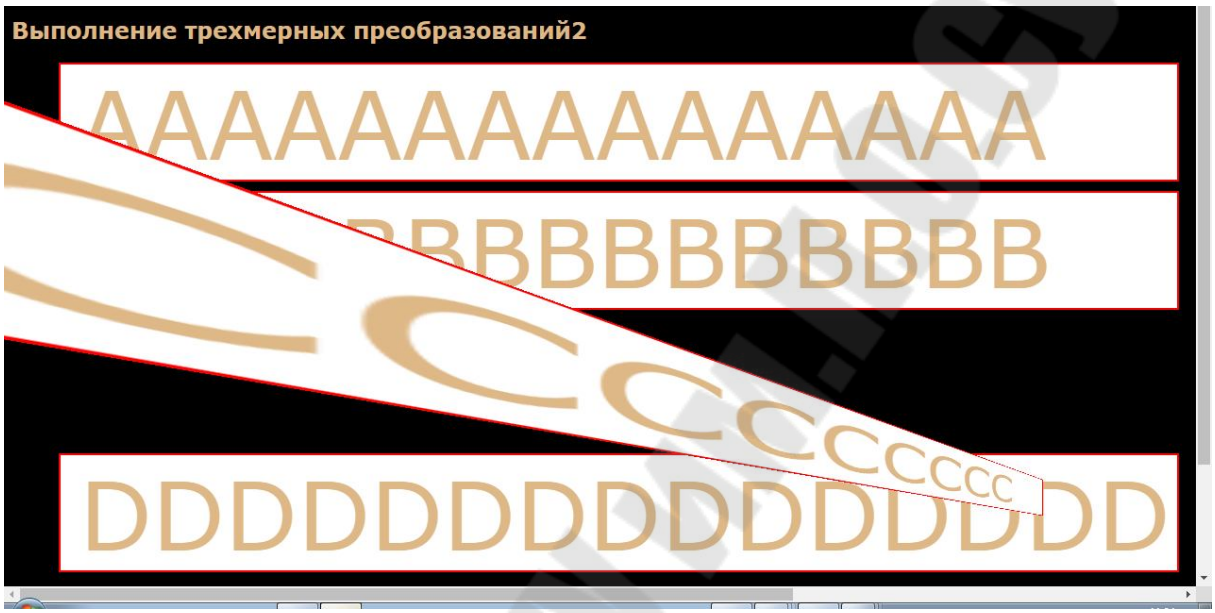


Рисунок 4.6 – Применение свойства perspective-origin

Стиль 3D-преобразований transform-style

Свойство определяет, как вложенные элементы отрисовываются в трехмерном пространстве. Свойство должно использоваться вместе со свойством transform и задается только для вложенных элементов.

transform-style: flat – Значение по умолчанию. Все дочерние элементы отображаются плоскими в двухмерной плоскости блока-контейнера.

transform-style: preserve-3d – Располагает элементы в трехмерном пространстве.

Пример

html-код

```
<div class="container flat">
        <div class="d1">
                <div class="d2"></div>
        </div>
```

```
        </div>

        <div class="container peserve3d">
            <div class="d1">
                <div class="d2"></div>
            </div>
        </div>
```

Часть CSS-кода

```
.flat .d1, .flat .d2 {
        transform-style: flat;
}

.peserve3d .d1, .peserve3d .d2 {
        transform-style: preserve-3d;
}

@keyframes rotate {
        50% {
            transform: none;
        }
}
```

Рисунок 4.7 – Визуальное расположение дивов отличается в зависимости от значения свойства transform-style

Лекция 5. Анимация с использованием API Canvas

Элемент *canvas*, добавленный в HTML5, предназначен для создания графики с помощью JavaScript. Например, его используют для рисования графиков, создания фотокомпозиций, анимаций и даже обработки и рендеринга видео в реальном времени.

1. Синтаксис. Основные команды

```
<canvas id="canvas"></canvas>


var canvas = document.getElementById("canvas");
var ctx = canvas.getContext("2d");
ctx.fillStyle = "green";
ctx.fillRect(x, y, width, height);
ctx.arc(x, y, radius, startAngle, endEngle, clockwise);
ctx.fill();
ctx.beginPath();
ctx.strokeStyle = "green";
ctx.lineWidth = 5;
ctx.stroke();
```

Прокомментируем этот код.

1-я строка: создаем Canvas на HTML-странице;

2-я строка: сохраняем в переменной *canvas* ссылку на элемент Canvas;

3-я строка: получаем доступ к рисованию на холсте;

4-я строка: задаем цвет заливки;

5-я строка: определяем прямоугольник с координатами левого верхнего угла, а также шириной и высотой;

6-я строка: определяем дугу с началом в точках *x* и *y*, указываем какой угол начальный, какой конечный; последний параметр определяет направление рисования дуги – против часовой стрелки;

7-я строка: команда залить область;

8-я строка: начать путь (для рисования линией);

9-я строка: цвет линии;

10-я строка: толщина линии;

11-я строка: нарисовать линию.

2. Рисуем две пересекающиеся наклонные линии

ctx.beginPath();

ctx.lineWidth = 5;

ctx.moveTo(10, 10); //передвижение без рисования линию до точки (10, 10)

ctx.lineTo(400, 400); //нарисовать линию до точки (400, 400)

ctx.moveTo(10, 400);

ctx.lineTo(400, 10);

ctx.stroke();

3. Рисуем оранжевый смайлик

ctx.strokeStyle = "orange";

ctx.lineWidth = 5;

ctx.beginPath();

*ctx.arc(225, 225, 200, 0, 2*Math.PI);*

ctx.moveTo(185, 190);

*ctx.arc(155, 190, 30, 0, 2*Math.PI);*

ctx.moveTo(325, 190);

*ctx.arc(295, 190, 30, 0, 2*Math.PI);*

Рисунок 5.1 – Рисование дугами

ctx.moveTo(365, 225);

ctx.arc(225, 225, 140, 0, Math.PI, false);

ctx.stroke();

4. Приступаем к движению

В примере, рассматриваемом ниже, квадрат движется, повинуясь нажатию клавиш со стрелками. Данный пример, кроме приема анимации, демонстрирует нам событийную модель JavaScript, а также показывает, как работать со «слушателем события».

Логика такова: функция *draw* – для отрисовки квадрата; функция *move* – уничтожает текущий квадрат путем очистки холста и рисует квадрат в новой позиции; функция *handler* – в случае, если произошло одно из четырех событий (четыре клавиши), выполняет функцию *move*; слушатель события прикреплен к элементу *body*. При нажатии на любую кнопку клавиатуры, *body* это слышит и выполняет функцию *handler*.

var draw = function(x, y) {

ctx.fillRect(x, y, 50, 50);

}

var move = function(speedX, speedY) {ctx.clearRect(0,0, can.width, can.height);

initX += speedX;

initY += speedY;

draw(initX, initY);

}

Прикрепляем слушатель.

body.addEventListener("keydown", handler);

function handler(event) {

if(event.keyCode == 37) {move(-speed, 0);}

```
if(event.keyCode == 38) {move(0, -speed);}
if(event.keyCode == 39) {move(speed, 0);}
if(event.keyCode == 40) {move(0, speed);}
}
```

(37, 38, 39, 40 – коды клавиш со стрелками, соответственно, «влево», «вверх», «вправо», «вниз»).

5. Простая анимация

Сделать анимацию из элемента на холсте HTML5 достаточно просто. Для этого устанавливается таймер, который постоянно вызывает элемент, обычно 30 или 40 раз в секунду. При каждом вызове код полностью обновляет содержимое всего холста. Если код написан правильно, постоянно сменяющиеся кадры сольются в плавную, реалистичную анимацию.

JavaScript предоставляет два способа для управления этим повторяющимся обновлением содержимого холста:

1) ***Функция setTimeout()***

Эта функция дает указание браузеру подождать несколько миллисекунд, а потом исполнить фрагмент кода, в данном случае код для обновления содержимого холста.

Синтаксис

var timerId = setTimeout(func / code, delay);

Отмена исполнения – *clearTimeout().*

Функция setTimeout возвращает числовой идентификатор таймера timerId, который можно использовать для отмены действия.

Синтаксис

var timerId = setTimeout(func / code, delay);

clearTimeout(timerId);

2) ***Функция setInterval()***

Метод setInterval имеет синтаксис, аналогичный setTimeout.

var timerId = setInterval(func / code, delay);

Смысл аргументов – тот же самый. Но, в отличие от setTimeout, он запускает выполнение функции не один раз, а регулярно повторяет её через указанный интервал времени. Остановить исполнение можно вызовом *clearInterval(timerId)*.

Пример 1. Создание прямоугольников, рождающихся сверху холста в произвольной позиции и падающих вниз с произвольной скоростью

Рисунок 5.2 – Падающие вниз квадраты

1) Определяем класс Rect (можно было бы и не создавать класс, но поскольку фигур у нас будет много, то создание шаблона – хорошее решение).

var Rect = function() {

this.x = Math.round(Math.random()(can.width-size));*

*this.y = -Math.round(Math.random()*20);//квадраты должны появляться выше холста – этим объясняется знак «минус»*

*this.speed = Math.floor(Math.random()*10) + 1;*

};

Вновь созданный объект класса Rect будет иметь случайную координату по x в пределах ширины холста. Начальная координата по y будет такой, что бы объект сначала был не виден (находится сверху холста). Скорость объекта задается в пределах от 1 до 11.

2) Функция draw будет заливать прямоугольник случайным цветом.

```
Rect.prototype.draw = function() {
var color1 = Math.floor(Math.random()*255);
var color2 = Math.floor(Math.random()*255);
var color3 = Math.floor(Math.random()*255);


ctx.fillStyle = "rgb(" + color1 + "," + color2 + "," + color3 + ")";
ctx.fillRect(this.x, this.y, size, size);
 };//end Rect.draw
```

3) Создание метода *move* у класса *Rect*

```
Rect.prototype.move = function() {
        ctx.clearRect(this.x, this.y, size, size);//перед тем как передвинуть //квадрат, нужно очистить место, которое он занимает в текущий момент
        this.y += this.speed;
};
```

4) Применение *setInterval*

```
setInterval(newRect, 10);
function newRect() {
        count++;
        info2.value = count;//счетчик созданных объектов
        var rect = new Rect();
```

```
    var index = setInterval(go, 50);
    function go() {
        rect.move();
        rect.draw();
    }
}
```

Отметим, что при такой реализации, все наши созданные объекты, даже те которые давно «упали» далеко ниже края холста, все равно «живут». В дальнейшем, мы будем уничтожать объекты, которые выходят за край холста.

Пример 2. Создание прямоугольников, рождающихся в центре холста и разлетающихся в стороны

Рисунок 5.3 – Квадраты, вылетающие из центра

1) Изменение в классе Rect

A. Теперь объекты создаются не сверху, а рождаются в центре холста

B. Теперь у них две скорости: по оси X и по оси Y

var Rect = function() {

```
        this.x = (can.width-size)/2;
        this.y = (can.height-size)/2;
        this.speedX = Math.floor(Math.random()*20) - 10;
        this.speedY = Math.floor(Math.random()*20) - 10;
};
```

2) Изменение в методе move

```
Rect.prototype.move = function() {
        ctx.clearRect(this.x, this.y, size, size);
        this.x += this.speedX;
        this.y += this.speedY;
};
```

6. Работа с текстом в Canvas

Контекст рендеринга canvas предоставляет два метода для рисования текста:

fillText(text, x, y [, maxWidth])

- Вставляет заданный текст в положении (x, y). Опционально может быть указана максимальная ширина.

strokeText(text, x, y [, maxWidth])

- Вставляет контур заданного текста в положении (x, y). Опционально может быть указана максимальная ширина.

Можно задать размер, стиль и шрифт текста через свойство font. В примере задаётся только размер и шрифт. Можно добавить наклон и жирность в начале строки.

Два последних аргумента fillText (и strokeText) задают позицию, с которой начинается текст. По умолчанию это начало линии, на которой «стоят» буквы – не считая свисающих частей букв типа p и y. Можно менять позицию по горизонтали, задавая свойству textAlign

значения "end" или "center", а по вертикали – задавая textBaseline "top", "middle", или "bottom".

Рисунок 3.4 – Создание текста на холсте

Пример1. fillText, strokeText

ctx.font = "48px Arial";

ctx.fillText(“Text Text Text", 10, 50);

ctx.font = "120px Arial";

ctx.strokeStyle = "white";

ctx.strokeText("HELLO", 10, 200);

Пример 2. Движущийся текст. Реализация эффекта подбора пароля

Рисунок 5.5 – Анимация текста

Из букв и восклицательного знака формируется массив. При каждом вызове функции, являющейся аргументом функции setInterval, на основе исходного массива, случайным образом формируется новый массив. Этот массив склеивается в слово. Как только это слово совпадет со словом «HELLO!», вызов функции прекращается. При этом в силу того, что каждый раз слово отрисовывается в случайной позиции, создается эффект дрожания или плавания слова.

Приводим код полностью, чтобы читатель мог отследить все нюансы.

```
window.onload = init;
function init() {
    var can = document.getElementById("canvas");
    var but1 = document.getElementById("button1");
    var but2 = document.getElementById("button2");
    var but3 = document.getElementById("button3");
    var ind;
    but1.addEventListener("click", clickHandler);
    but2.addEventListener("click", clickHandler2);
    but3.addEventListener("click", clickHandler3);
```

(Создали холст, кнопки и навесили на них слушатели)

```
    can.width = 480;
    can.height = 200;
    var ctx = can.getContext("2d");
    ctx.fillStyle = "yellow";
    ctx.font = "48px Arial";
    ctx.fillText("Text Text Text", 10, 50);
```

(Параметры текста)

```
    function clickHandler() {
```

```
ctx.font = "120px Arial";
ctx.strokeStyle = "white";
ctx.strokeText("HELLO", 10, 180);
}
```

(Нажатие на первую кнопку)

```
var a = ["H", "E", "L", "L", "O", "!"];
function randomWord() {
    ctx.clearRect(0, 0, can.width, can.height);
    ctx.font = "120px Arial";
    ctx.strokeStyle = "white";
    var word = "";
    var b = [];
    for(var i = 0; i < a.length; i++) {
        b.push(a[i]);
    }
    var len = b.length;
    for(var i = 0; i < len; i++) {
        var index = Math.floor(Math.random()*(len-i));
        word += b[index];
        b.splice(index, 1);
    }
    if(word == "HELLO!") {
        clearInterval(ind);
    }
    ctx.strokeText(word, 20, 150);
```

```
        return word;
    } //конец функции randomWord
```

Функция randomWord возвращает сформированное слово из букв, помещенных в массив *a*.

Функция randomWord2 отличается от первой добавленным эффектом дрожания слова.

```
    function randomWord2() {
        //здесь код повторяется
        var x = Math.round(Math.random()*20) - 10;
        var y = Math.round(Math.random()*20) - 10;
        ctx.strokeText(word, 20+x, 150+y);
        return word;
    }
```

Ниже выписан код функций, выполняющихся при нажатии на оставшиеся две кнопки.

```
        function clickHandler2() {
            ind = setInterval(randomWord, 10);
        }
        function clickHandler3() {
            ind = setInterval(randomWord2, 10);
        }
    }// конец функции init
```

7. Работа с объектом Image в Canvas

В компьютерной графике проводится различие между векторной и растровой графикой. Естественно рисовать на холсте примитивами очень неудобно и требует определённых трудозатрат, и резуль-

тат иногда явно хромает качеством. Поэтому естественно в canvas api предусмотрено взаимодействие с изображениями. Добавление изображения условно можно разделить на два шага: создание JavaScript объекта Image, а второй и заключительный шаг это отрисовка изображения на холсте при помощи функции drawImage. Рассмотрим оба шага подробнее.

1) Создание нового графического объекта:

```
var picture = new Image();
 picture.src = 'img.png';
```

2) Рисование изображения на холсте:

```
picture.onload = function() {
        ctx.drawImage(picture, x, y);
}
```

Пример 1. Смайлик, бегающий по картинке вправо влево

```
var img2 = new Image();
img2.src = "../img/smile.png";
img2.onload = function() {
setInterval(move, 20);
function move() {
        ctx2.clearRect(0, 0, w, h);
        if(flag) {
                ctx2.drawImage(img2, x+=5, 180);
                if(x > 500) flag = false;}
        if(!flag) {
                ctx2.drawImage(img2, x-=5, 180);
                if(x < 0) flag = true;}
}};
```

Изменим функцию move, так чтобы смайлик, отталкивался не только от левого и правого края холста, но и от верхнего и нижнего. Сделаем так, чтобы при каждом столкновении с краем холста, скорость смайлика случайным образом изменялась.

```
function move() {
ctx2.clearRect(0, 0, w, h);
ctx2.drawImage(img2, x+=speedX, y+=speedY);
if(x >= 500 || x <= 0) {
        speedX = -speedX + Math.round(Math.random()*21);}
if(y >= 380 || y <= 0) {
        speedY =-speedY + Math.round(Math.random()*2-1);}
}
```

Рисунок 5.6 – Смайлик движущийся поверх картинки

Функция *.drawImage* с девятью параметрами

```
drawImage(img, x1, y1, width1, height1, x2, y2, width2, height2);
img.onload = function() {
```

```
        ctx.drawImage(img, 10, 10);

        ctx2.drawImage(img, 250, 350, 300, 300, 30, 30, 100, 100);

    };
```

Эта функция копирует из картинки img нужный кусочек с параметрами *x1, y1, width1, height1* и помещает его в новую позицию *x2, y2*, производя при этом масштабирование.

Бельский Вадим Алексеевич

АНИМАЦИОННАЯ ГРАФИКА

Пособие
по одноименной дисциплине
для слушателей специальности переподготовки
1-40 01 77 «Web-дизайн и компьютерная графика»
заочной формы обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 14.01.19.
Рег. № 80Е.
http://www.gstu.by