Gimp

- Плагины
- Кисти (paintbrush)
- Мультипликация и SVG

???????

Назначение плагинов

Название	Описание
Resynthesizer	Удаление объектов с изображения с восстановлением фона.
Layer-via-copy	Ускорение перемещения объектов на новый слой.
Nik Collection Free	Набор из 7 плагинов для ускорения обработки изображений от Google. Установка: • установить дистрибутив в директорию C:\Program Files\Google\Nik Collection (по умолчанию) • распаковать скрипты отображения плагинов в меню и скопировать файлы *.py в директорию %appdata%\GIMP\2.10\plug-ins • В меню Filters -> Photography появятся установленные плагины.

????? (paintbrush)

Цвет определяется основным цветом (Foreground color).

Статические параметры кисти

Параметры, определяющие реальное отображение на листе.

Opacity	Прозрачность кисти. Установка в 50 при RGB 000 соответствует начальному цвету RGB 73 73 73.
Размер (Size)	удерживая Alt, изменяется колесиком мыши.
Aspect ratio	Соотношение между горизонтальным и вертикальным размером кисти. Актуально не для всех кистей.
Angle	Угол поворота кисти
Spacing	Расстояние следующего элемента при перемещении с удержанием
Hardness	Толщина размывания краев. Актуальна при работе с планшетом, для мыши необходимо выставлять.
Force	Стартовая толщина слоя, при проведении заново (при удерживании кнопки), добавляется толщина. То есть, при однократном проведении цвет будет мягче. Максимум - до значения Opacity.

Динамические параметры кисти

Параметры, определяющие как будут изменяться статические параметры кисти в зависимости от действий. Настройка в виде матрицы, где строки - статические параметры, столбцы - динамические параметры.

Pressure	Зависимость от давления. Актуально при работе с планшетом
Velocity	Зависимость от скорости перемещения курсора
Direction	Зависимость от направления перемещения курсора
Tilt	Зависимость от угла наклона перемещения курсора
Wheel/Rotation	Зависимость от перемещения колесика мыши
Random	Зависимость от случайного числа
Fade	Зависимость от степени прозрачности изображения

Приемы работы.

Прямая линия, любое направление: однократное нажатие (нажать и отпустить) ЛКМ, нажать и удерживать Shift переместить в нужную точку и однократное нажатие (нажать и отпустить) ЛКМ.

Прямая линия, шаг угла наклона 15 градусов: однократное нажатие (нажать и отпустить) ЛКМ, нажать и удерживать Shift+Ctrl, переместить в нужную точку и однократно нажать (нажать и отпустить) ЛКМ. Используется чаще для строгих горизонтальных и вертикальных линий.

?????????????? ? SVG

Принципы мультипликации

Сжимать и растягивать
В зависимости от того, из чего что-либо сделано, объекты деформируются при движении. Сжимание и растягивание создает иллюзию веса и объема объекта
Предвкушение
Пользователи могут не понять анимацию, если в ней нет последовательности действий, которая четко ведет от одного действия к следующему. Они должны предвидеть изменения до того, как они действительно произойдут. В мультфильмах это движение подготавливает зрителя к важному действию, которое собирается выполнить персонаж, например, сгибание колена перед прыжком. В Интернете это может быть нажатие кнопки перед началом более масштабной анимации, которая запускается при нажатии кнопки. Мысленно направляйте своих пользователей к тому, на чем им следует сосредоточиться, прежде чем приступать к основному эффект, особенно если важно начало анимации.
Постановка
Постановка направляя внимание пользователя на такие действия, как небольшой джиггле призыв к действию кнопки. Постановка помогает провести пользователей через рассказываемую историю или идею: например, через этапы процесса оформления заказа.
Прямолинейное действие и поза к позе
Существует два основных подхода к анимации на большом экране. При прямолинейном подходе аниматор начинает с первого рисунка и работает от рисунка к рисунку до конца сцены. От позы к позе аниматор рисует основные точки в анимации и позже создает (или поручает помощнику создать) промежуточные точки. Хотя, казалось бы, этот принцип применим только к раскадровке, он также связан с рисованием ключевых кадров и тем, как анимация заполняет пространство или время между ними.
Последующее и перекрывающееся действие
Последующее действие - это включение дополнительного движения по завершении основной анимации. Например, если персонаж бежит и останавливается, ее волосы и одежда, скорее всего, подпрыгивают и возвращаются на место после того, как ее ноги и тело перестают двигаться, догоняя основную массу персонажа. Ничто не останавливается сразу. Перекрывающееся действие - это когда некоторые компоненты слегка задерживаются после того, как другие компоненты меняют направление, например, как у Wile E. Ноги Койота продолжают двигаться вперед, когда он падает со скалы. Его ушам требуется мгновение, чтобы последовать за ним. Если ваша CSS-анимация когда-нибудь станет достаточно сложной, чтобы потребовать последующих
и накладывающихся друг на друга действий, выбор времени будет иметь решающее значение для того, чтобы ваши эффекты заработали.
Замедляйте вход и выход
Точно так же, как автомобили не заводятся и не останавливаются на полной скорости — скорее, они разгоняются от остановки до полной скорости и снова сбрасывают скорость до нуля. медленные входы и выходы делают анимацию более реалистичной и смягчают

действие. Только механическая анимация будет выполняться с линейной скоростью.
Принцип медленного ввода и
замедленного вывода гласит, что начало и конец анимации более интересны, чем ее середина; и поэтому, если анимация не механическая, анимация должна выполняться быстрее всего в середине анимации, с более медленное начало и более медленный конец. В мультяшной анимации эффект создается за счет большего количества ячеек на концах и меньшего количества в середине действия. В CSS этот эффект создается за счет настройки функций синхронизации кубического Безье на что-то отличное от линейного

Дуги

Принцип дуг гласит, что почти все действия выполняются по дуге или слегка круговой траектории. Представьте, что ваша рука движется взад и вперед во время ходьбы: ваша рука описывает дугу взад и вперед, а не всегда остается на равном расстоянии от земли. Линейная анимация может быть очень механической. Дуги могут быть созданы с помощью детального управления в анимации ключевого кадра. Из-за этого разработчики CSS часто используют анимацию CSS вместо переходов; анимации обеспечивают большую детализацию при создании дугообразного контура, в то время как переходы CSS позволяют перемещаться только между двумя состояниями. Однако с некоторыми функциями синхронизации кубического Безье создание дуги с помощью CSS-переходов на самом деле не только возможно, но и довольно просто.

Второстепенное действие

Второстепенные действия могут обогатить основное действие, добавляя измерение, дополняя или усиливая основное действие и придавая сцене больше жизни. Если вы включаете второстепенные действия, анимации должны работать сообща, поддерживая друг друга. Например, если ваша основная анимация перемещает модуль на страницу, вторичным действием может быть нажатие основной кнопки с призывом к действию внутри модуля, которая опускается на место, а затем завершает свое действие немного позже, чем основной модуль завершил анимацию. Второстепенное действие должно подкреплять основное. Допустимо буквально “мыслите нестандартно” и анимируйте дочерний элемент иначе, чем его родительский.

Синхронизация

Хронометраж, вероятно, является самым важным из принципов. В то время как в традиционной анимации это основано на количестве кадров, в CSS-анимации это больше связано с созданием соответствующего количества времени для считывания движения, но не настолько долгого, чтобы сайт казался медленным. Хронометраж включает в себя не только продолжительность анимации , но также задержку и функцию синхронизации. Когда дело доходит до хронометража, правильных ответов нет. Знание хронометража приходит с опытом и экспериментами, если оно вообще приходит. Я рекомендую использовать метод проб и ошибок, чтобы уточнить хронометраж вашей анимации, а затем сократить время вдвое: хотя вы, возможно, хотите, чтобы ваша анимация продвигалась достаточно медленно, чтобы оценить разницу во времени и определить наилучшее сочетание продолжительности, задержки и прогрессии, вы не хотите, чтобы ваш сайт выглядел медленным.

Преувеличение

Преувеличение - это выделение движений за пределами их естественного состояния, чтобы привлечь внимание к тому, на чем вы хотите, чтобы пользователь сосредоточился. Небольшое преувеличение может придать анимации дополнительную жизнь и фактически

сделать ее более реалистичной. Используйте хороший вкус и здравый смысл: преувеличение - это не крайнее искажение, а скорее небольшое искажение, которое придает акцент, не будучи настолько преувеличенным, чтобы быть видимым для ваших посетителей.

Сплошное рисование

Принцип сплошного рисования включает в себя принципы рисования или кодирования форм, которые создают иллюзию трехмерности, с помощью веса и сплошной формы. В CSS это включает использование прямоугольных теней, градиентов и преобразований, придающих вашему контенту иллюзию трехмерности. Если вы не занимаетесь 3D-анимацией или спрайтингом, этот принцип имеет лишь косвенное отношение к веб-анимации, как в Web, мы рисуем с помощью CSS в двумерном пространстве.

Обращение

Принцип привлекательности связан с харизмой, правдоподобием и интересом. Привлекательность в Интернете включает в себя легко читаемый дизайн, четкую прорисовку и движение, которые захватят и заинтересуют посетителя. Анимация должна быть привлекательной как для ума, так и для глаз.

12 кадров

При покадровой аннимации - обычно 12 кадров/сек хватает.

CSS аннимация

Ресурсы:

transitions and animations in css +estelle
pro css3 annimation Dudley Weyl

12 принципов аннимации

Методы:

2 режима: трансляция и аннимация
Параметры трансляции:

---

transition-property - изменяемое свойство (или список разделенный запятыми). Не перечисленные свойства изменяются мгновенно.

all - изменить все изменяемые свойства с одинаковой скоростью
можно разделить скорости изменения:

```
transition-property: all, border-radius, opacity;
transition-duration: 1s, 2s, 3s;
```

---

transition-duration - время изменения. ms, s. Если указано в обоих точках - указывается для точки назначения. Если указано в одном месте (любом) - используется для обоих преобразований.

```
div p {background-color: rgba(0, 0, 0, 0);transition-property: background-color;transition-duration: 2s;}
div p:hover {background-color: rgba(0, 0, 0, 0.1);transition-duration: 200ms;}
```

В этом случае при переходе в hover 200ms, обратно - 2s

---

transition-timing-function Также можем указать различные функции для разных свойств.

ease	Функция по умолчанию, переход начинается медленно, разгоняется быстро и замедляется в конце. Соответствует cubic-bezier(0.25,0.1,0.25,1).
linear	Переход происходит равномерно на протяжении всего времени, без колебаний в скорости. Соответствует cubic-bezier(0,0,1,1).
ease-in	Переход начинается медленно, а затем плавно ускоряется в конце. Соответствует cubic-bezier(0.42,0,1,1).
ease-out	Переход начинается быстро и плавно замедляется в конце. Соответствует cubic-bezier(0,0,0.58,1)
ease-in-out	Переход медленно начинается и медленно заканчивается. Соответствует cubic-bezier(0.42,0,0.58,1)
cubic-bezier(x1, y1, x2, y2)	Позволяет вручную установить значения от 0 до 1 для кривой ускорения. На этом сайте вы сможете построить любую траекторию перехода
initial	Устанавливает значение свойства в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Еще варианты функций

---

transition-delay Также может быть несколько заданных параметров. Советуют 50ms.

```
nav li ul {
  transition-property: transform;
  transition-duration: 200ms;
  transition-timing-function: ease-in;
  transition-delay: 50ms;
  transform: scale(1, 0);
  transform-origin: top center;
}
```

```
nav li:hover ul {
  transform: scale(1, 1);
}
```

transitionend - событие при завершении преобразования, только для изменяемых свойств. Можем добавить в события. 3 атрибута. Только если преобразование успешно завершилось.

propertyName - имя CSS свойства
pseudoElement - элемент, на котором произошло событие
elapsedTime - время преобразования, обычно transition-duration

```
document.querySelector('div').addEventListener('transitionend',
  function (e) {
  console.log(e.propertyName);
});
```

---

transition: all 200ms ease-in 50ms; - короткая запись 4 параметров

```
nav li ul {
  transition: transform 200ms ease-in 50ms,
      opacity 200ms ease-in 50ms;
  ...
}
nav li ul {
  transition: all 200ms ease-in 50ms;
  ...
}
nav li ul {
  transition: 200ms ease-in 50ms;
  ...
}
```

animation - укороченная запись : none | <series of individual animation properties> <animation-duration> || <animation-timing-function> || <animation-delay> || <animation-iteration-count> || <animation-direction> || <animation-fill-mode> || <animation-play-state> || <animation-name>

Создание кейфрейма

```
@keyframes animation_identifier {
  keyframe_selectorA {
    property1: value1a;
```

```
    property2: value2b;
  }
  keyframe_selectorB {
    property1: value1b;
    property2: value2b;
  }
}
```

Добавление аннимации к элементу

```
div {
     animation-name: change_bgcolor;
 animation-duration: 2000ms;
 }
 Могут перечисляться через запятую
 animation-iteration-count - количество повторов
число, infinite
 animation-direction - направление аннимации
normal от 0 до 100
reverse от 100 до 0
alternate первая от 0 до 100 вторая от 100 до 0 и т д
alternate-reverse наоборот
 animation-delay
 .rainbow {
    animation-name: red, orange, yellow, blue, green;
    animation-duration: 1s, 3s, 5s, 7s, 11s;
    animation-delay: 3s, 4s, 7s, 12s, 19s;
}
```

Последовательность аннимации

Сочетание задаржки и последовательности может создать эффект перехода одной аннимации в другую, когда следующая начинается точно после завершения предыдущей. Данный подход может плохо работать на медленных браузерах. (например одновременный старт нескольких аннимаций, хотя они должны были запуститься последовательно) Вариант - js события завершения предыдущей аннимации.

```
document.querySelectorAll('li')[0].addEventListener( 'animationend',
  function(e) {
      document.querySelectorAll('li')[1].style.animationName = 'orange';
  },
```

```
  false );
```

Функция аннимации

animation-timing-function аналогично transition
step-start, step-end, steps() - шаговые функции. step-start = steps(1, start) - пропускаем 0%, step-end пропускает 100%
animation-direction аналогично transition
animation-play-state: running | paused
animation-fill-mode: none | forwards | backwards | both
определяет воздействие аннимации на элемент, т е что будет потом с элементом. Размер воздействия определяется количеством завершенных итераций.
none Значения в 0% применяются только после старта, delay на значениях по-умолчанию. После annimationend (не 100%, может закончиться раньше) возвращаются на умолчания.
forwards Применяются сразу же, после annimationend умолчания.
backwards после annimationend остаются
both forwards+backwards

События аннимации animationstart, animationend, animationiteration

Спрайты

Спрайт - склеенная картинка для реализации движения.
50 на 50, 8 изображений

```
@keyframes bulkinwather {
                from {
                    background-position: 0 0;
                }
                to {
                    background-position: -400px 0;
                }
            }
            .mysprite {
                height: 50;
                width: 50;
                background-image: url('jpg/sprite.gif');
                animation-name: bulkinwather;
                animation-duration: 1s;
                animation-timing-function: steps(8, end);
                animation-delay: 1s;
                animation-iteration-count: infinite;
            }
```

Функции аннимации можно менять в кейфреймах, за счет этого получим неравномерное движение. Но изменяемые параметры должны быть в одном блоке с функцией аннимации. Если задать пустой блок только с функцией аннимации - влияние ни на что не будет оказано.

SVG

```
<html>
<body>
<svg x="0px" y="0px" width="450px" height="100px" viewBox="0 0 450 100">
 <rect x="10" y="5" fill="white" stroke="black" width="90" height="90"/>
 <circle fill="white" stroke="black" cx="170" cy="50" r="45"/>
 <polygon fill="white" stroke="black" points="279,5 294,35 328,40 303,62
  309,94 279,79 248,94 254,62 230,39 263,35"/>
 <line fill="none" stroke="black" x1="410" y1="95" x2="440" y2="6"/>
 <line fill="none" stroke="black" x1="360" y1="6" x2="360" y2="95"/>
</svg>
<svg viewBox="0 0 218.8 87.1">
 <g fill="none" stroke="#000">
   <path d="M7.3 75L25.9 6.8s58.4-6.4 33.5 13-41.1 32.8-11.2 30.8h15.9v5.5s42.6
     18.8 0 20.6" />
   <path d="M133.1 58.2s12.7-69.2 24.4-47.5c0 0 4.1 8.6 9.5.9 0 0 5-10 10.4.9 0
     0 12.2 32.6 13.6 43 0 0 39.8 5.4 15.8 15.4-13.2 5.5-53.8
     13.1-77.4 5.9.1 0-51.9-15.4 3.7-18.6z" />
 </g>
</svg>
</body>
</html>
```

svg
    x, y - координаты угла начала
    width, height - ширина, высота изображения
    viewBox - отображаемая часть изображения
rect
    fill="white" - заполнение
    stroke="black" - границы
g - объединение нескольких элементов, заливка и границы распространяются на потомков
path - путь
    M,m M7.3 - переместить стартовую позицию в точку
    L,l - LineTo
    H,h - horizontal line
    V,v - vertical
    z - go to begin

C,c - cubic bezier
S,s - че-та
Q,q - quadro bezier
T,t -
A,a - Elliptic arc