Twine 2. Обзор и руководство для начинающих

По просьбам трудящихся и своей инициативе – разбор Twine 2, инструмента для написания Интерактивной литературы и текстовых квестов.

==========

Об Twine

Twine 2 – это идейный потомок проекта Twee/TweeBox. В свое время существовал проект Twee, написанный на языке Python. Его внутренняя система разметки/макросов больше всего напоминала Wiki-разметку. Насчет популярности не скажу, но своя армия фанатов у него была. Достаточно большой проблемой Twee была ее скорость работы. Поскольку движок был написан на Python, для сборки десктопных приложений использовалась библиотека **wxWidgets**, известная своей невысокой скоростью работы.

Позже был создан TweeBox - порт, переписанный на JavaScript. А потом Twee был переименован в Twine. Возможно, в то время и отпочковался коммерческий проект AXMA, более известный у нас, нежели ее прародитель. Потому что сделан и продвигается отечественной командой разработчиков.

Почему не AXMA?

К слову, AXMA сохранила тормознутость предшественника. Кто-то говорит, что все нормально, но у меня она всегда тормозит. А еще меня отталкивает жопоголизм разработчиков: за 1000 рублей нам предлагают сохранение HTML-файла, отсутствие копирайта AXMA в файле и (sic!) лицензия на редактирование файла. Это как если бы Adobe требовали ставить копирайт "powered by PhotoShop" в каждой

отфотошопленной фотке. И меня пугает внутренний язык разметки/макросов. Но тут кому как.

Сейчас Twine полностью переписан на JavaScript (jQuery/Backbone/Underscore), вместо wxWidgets использует среду исполнения node-webkit. Это дало ей преимущество в скорости. И возможность делать самые разные вещи, которые мы рассмотрим, когда будем собирать готовое приложение для публикации.

Внутри Twine 2 есть такие языки как Harlowe и SugarCube (старый wiki-образный код + ч**о**рное оформление). Так же присутствует SnowMan, который предлагает использовать вместо макросов чистый JS внутри тэгов вида <%= %>.В целом, для людей, работавших с AXMA или писавших свою IF-книгу на чистом HTML, проблем возникнуть не должно.

Об Harlowe

Harlowe – достаточно простой и логичный язык макросов, встроенный в Twine 2. Писать на нем лично мне оказалось гораздо проще, удобнее и понятнее, чем в AXMA. Я "верующий" программист, и верую в то, что язык должен отвечать задачам. А писать скрипты в Wiki-разметке это содомия. Прочитайте про философию **MVC** на досуге.

Технические подробности

Внешне Harlowe выглядит совместным потомством языков Clojure и Python. От первого ему достались команды, обернутые в круглые скобочки и минимальный набор используемых знаков пунктуации; а от второго он получил наименование переменных через $variable, три вида массивов и отсутствие "логических скобок" в некоторых макросах. Хотя лично я бы сделал Harlowe более питонообразным. Потому что макрос вида (if (a>b): (print "a>b")) выглядит изящнее чем

(if: (a>b))[(print: "a>b")]. Как минимум, меньше вложенных скобочек и в первом случае я вижу логику.

Самое простое в Twine – это создание ссылок на параграфы. Это делается так: [[текст ссылки->имя параграфа]] или [[имя параграфа<-текст ссылки]] или просто [[имя параграфа]]. Пробелы вокруг имени параграфа старайтесь не ставить, потому что может начаться всякая неожиданная хрень.

Еще в Harlowe есть inline-оператор {}. Фигурные скобки позволяют превратить несколько абзацев текста в одну строку и убрать лишние пробелы. Очень удобно, потому что Twine понимает переносы строк внутри сложных макросов как переносы в тексте.

Из разметки полезно запомнить *Наклонный текст*, **Жирный текст** и ^^Верхний индекс^^.

Макросы

(set: $variable to value) – устанавливает некоторое значение в переменной. Имя переменной начинается на $.

(print:) – выводит некоторый текст или результат вычислений. Например (print: 5 + 5) выведет 10.

(display: "имя параграфа") – выводит содержимое параграфа.

(if:)[](else-if:)[](else:)[] – стандартная логическая цепочка if-else. В квадратных скобках пишутся действия, которые выполняются при условиях. В (if:) и (else-if:) после двоеточия пишется условие, в(else:) - нет. конструкций (else-if:) может быть несколько. В целом, такая структура соответствует операторам switch-case-dafault в том же JavaScript.

(a:) – создает нумерованный массив. Элементы массива указываются после двоеточия через запятую: (a: 1,2,3,4,5).

(datamap:) - создает именованный массив вида ключ-значение. Т.е. (datamap: "Name1", Value1, "Name2", Value2) вернет массив строк Name1 и Name2, которым соответствуют значения Value1 и Value2.

Макрос (datanames:) вернет массив имен в datamap, а (datavalues:) - список значений.

(current-date:), (current-time:), (monthday:) и (weekday:) возвращают Текущую дату, Текущее время, Число месяца и День недели.

(link: "text")[New text] - создает интерактивную ссылку, после нажатия на которую текст ссылки подменяется на содержимое квадратных скобок.

(link-reveal: "text")[after text] - создает ссылку, по нажатию на которую текст ссылки дополняется текстом из квадратных скобок.

(link-goto: "text", "passage") - полный аналог [[text->passage]].

(link-goto: "passage") аналогичен [[passage]].

(go-to: "passage") – безусловный переход на блок passage.

(alert: "Message") - выводит сообщение.

(confirm: "Message") - выводит сообщение с кнопками ОК и ОТМЕНА. Возвращает true или false. Использует Python-подобный принцип: (set: $wantCake to (confirm: "Do you want a cake?")) – и позволяет напрямую забирать ответ пользователя.

(prompt: "Message") - выводит сообщение и поле для ввода. Можно использовать для запроса пользовательского имени: (set: $name to (prompt: "Your name, please:")).

Это самые основные макросы, которые понадобятся в самом начале. Остальное есть в документации по Harlowe.

[Инструкция к Twine 2]

Интерфейс Twine довольно прост. Его можно разделить на главное окно и окно истории. На главном окне вы видите список своих

историй, открываются они двойным кликом. В меню справа полезные кнопки это **+ История**, **Форматы**, и **Язык**. В меню Форматы можно выбрать между форматами Harlowe, Snowman и SugarCube и выставить его по умолчанию. Harlowe стоит по умолчанию, собственно о нем и идет здесь речь.

После нажатия на **+ История** нам предлагают ввести имя истории и **Добавить** ее или **Отменить** решение. История появится в левом верхнем углу списка. На самом блоке истории отображается миниатюра истории. В правом верхнем углу миниатюры под значком шестеренки скрываются подменю: **Запустить историю**, **Тестировать историю**, **Опубликовать в файл**, **Переименовать историю**, **Сделать дубликат истории**. Чтобы открыть историю для редактирования по ней нужно кликнуть два раза.

В окне истории вы видите синее размеченное поле и меню внизу. Там расположены кнопки **Домой** в виде домика, **Кнопка с названием истории**, **поисковая строка**, кнопки масштаба,**Тестировать**, **Запустить** и **+Параграф**. В меню с названием истории есть разделы **Редактировать JavaScript**, **Редактировать таблицу стилей**, **Сменить формат истории**, **Переименовать историю**, **Привязать к сетке**, **Статистика истории**, **Получить копию для вычитки** и **Опубликовать в файл**. Лично для меня полезными были Редактировать JavaScript (учтите, скрипт загрузится в самом начале, до того, как загрузился движок, на котором работает сама игра), Редактировать таблицу стилей, Привязать к сетке (очень удобная вещь), получить копию для вычитки и Опубликовать в файл (открывается окно для сохранения истории в standalone-файл с движком внутри).

Чтобы добавить параграф нужно нажать **+ Параграф**. На поле появится новый блок. У блока есть всплывающее меню из четырех иконок: **Удалить**, **Изменить**, **Тестировать** и **Сделать "имя**

параграфа" отправной точкой истории. Параграф также можно удалить кнопкой Delete, а открыть двойным кликом.

У каждого параграфа есть название, на на которое ссылается указатель, а также тэги, которые можно использовать для стилизации и форматирования. В меню окна приложения есть пункты Twine и Редактировать. В меню Twine есть пункт Показать библиотеку – удобно для добавления историй, хотя это можно сделать через главное меню. Отменять/вырезать/копировать/вставлять можно и через Ctrl+Z/X/C/V. Да, правая кнопка мыши не работает.

Процесс создания игры

В качестве примера по созданию простой игры я буду переносить на Twine свою же игру Press [RE:START]. Исходники можно взять **здесь**, готовый Twine-файл со стилями **здесь**.

Для начала я создаю пустой параграф index и вставляю туда исходный текст первой страницы, который помещен внутрь <body>.

Поскольку Twine довольно превратно понимает [квадратные скобки], тест всех ссылок нужно обрамлять в **[ ]**. Однако, если ВСЕ ваши ссылки обрамляются таким образом, проще открыть Меню истории -> Редактировать таблицу стилей и добавить следующий блок: **`tw-link:before{content: "[";}tw-link:after{content: "]";}`**.

Далее, заменяю ссылки вида:[RE:START]на [[000<-RE:START]].

Twine автоматически создает параграфы, на которые были добавлены ссылки, поэтому проблема с созданием параграфов вручную отпадает сама собой.

По очереди переносим содержимое всех страниц в соответствующие блоки, заменяю ссылки на квадратные скобки. Поскольку лишний раз перетаскивать и переписывать мне было лень,

ссылки я оформлял в обратном стиле: [[имя параграфа<-текст ссылки]]. Потому что в HTML сначала идет href="", а потом текст ссылки. Я действительно ленив.

Поскольку мне не нужно возвращаться к предыдущей странице, я открываю в меню "редактировать таблицу стилей" и пишу туда: `tw-sidebar{display:none}`. Это позволяет скрыть сайд-бар, на котором показываются стрелочки. В принципе, в Twine можно вставлять код на JavaScript/jQuery, но он исполняется сразу при загрузке, а сами "страницы" генерируются динамически из текста пассажей. Поэтому полностью удалить сайд-бар или набить его новым функционалом через JS лично у меня пока не получается.

Как видите создание страниц довольно просто само по себе, расширять функционал можно через макросы, а некоторые рецепты и фишечки я покажу ниже.

Расширенный функционал

В Harlowe учитываются тэги, прикрепленные к блоку. Добавить их можно по кнопке **+Тэг** под названием истории.

Если вы хотите добавить шаблонный элемент, который будет на каждой странице, добавьте тэг **header** (содержимое будет отображать над текстом) или **footer** (содержимое будет отображать под текстом).

Тэг **startup** позволяет вставить макросы, которые загружаются до начала истории. Здесь можно, например, проинициализировать все переменные.

Также Harlowe поддерживает локальное хранилище в браузерах. Это позволяет сохранять и загружать состояния игры:

(savegame: "Slot A") - сохраняет игру в слот "Slot A"

(loadgame: "Slot A") - загружает игру в слоте "Slot A"

(saved-games:) - возвращает список сохранений.

В Harlowe есть свои особенности по обращению с переменным и массивами в частности. Так, можно напрямую обратиться к элементу массива через (a:)'s 1st, 2nd, 3rd или last. Так же вместо этого поддерживается синтаксис вида (a:)'s (1) для доступа к первому элементу, (a:)'s (-1) - к последнему элементу.

Twine 2, поскольку он написан на HTML/JS, поддерживает скрипты и HTML/CSS. Для кастомизации содержимого через CSS полезно знать следующие элементы внутренней разметки Twine 2:

- **tw-story** - корневой элемент истории
- **tw-passage** - элемент, в котором рендерится сама история. Содержит стандартные свойства, аналогичные элементу div, с той лишь разницей, что свойство width имеет значение по умолчанию 60%.
- **tw-link** - элемент, в который оборачиваются ссылки типа (link:), (goto:) и [[]]. Поддерживает класс .visited.
- **tw-sidebar** - элемент, в котором отображаются кнопки вперед и назад.
- **tw-icon** - иконки вперед и назад. Класс **.undo** отвечает за иконку назад, **.redo** - за иконку вперед.
- **mark** - элемент, используемый для части HTML, которую нужно оставить без изменений. по умолчанию имеет свойства **color:rgba(0,0,0,0.6); background-color: #ff9**.

Итого:

- Из плюсов: Twine 2 получился удобным, понятным и быстрым. Язык макросов Harlowe – простой, понятный и единообразный. А люди, знакомые с Python, вообще будут себя чувствовать как дома.
- Из минусов: сложность внедрения JavaScript (статьи предлагают прописывать функции через `window.имяФункции()`) и

невозможность написания собственных макросов. Что характерно, для SugarCube свои макросы писать можно.

- Из неожиданного: Twine мало известен в русском IF-сообществе, но получил распространение среди маргинальных и ЛГБТ-игроделов. Русская Twine-тусовка позиционирует себя как крайний андерграунд и анти-мейнстрим. Даже на фоне всего остального IF-сообщества. Подробнее – **здесь**.

Публикация готовой игры

Предельно простой вариант

В левом нижнем меню вашей истории выберите "Опубликовать файл" и укажите куда его сохранить. Twine сам соберет HTML файл и добавитв него код движка. Остается только опубликовать. Для этого у сообщества Twine есть сайт http://philome.la/. Или можете разместить ее в любом другом месте, на страничке уже есть код движка.

Node-WebKit

1. Экспортируйте историю, сохранив под названием index.html
2. Скачайте **Node-Webkit** (если хотите, можете скачать сборку не под вашу операционную систему, например для Linux)
3. Распакуйте его в папку с экспортированным файлом
4. Откройте package.json и добавьте туда:

```
{
    "name": "Имя истории",
    "description": "Описание истории",
    "main": "index.html",
    "window": {
        "show": false,
        "toolbar": false,
```

```
            "width": 1024,
            "height": 768
        }
}
```

5. Для проверки запустите nw.exe - должна запуститься ваша история. Можете переименовать nw.exe и заархивировать.

Рецепты

[Рецепты: Инвентарь]

Вариант первый

1. Создайте отдельный параграф startup и добавьте ему тег startup.
2. В блок вставьте следующий код:(set: $inventory to (a:)). Эта строка создает пустой массив для инвентаря.
3. Создайте блок inventory с тегом inventory и вставьте следующий код: `(text: $inventory.join("\n"))(link-repeat: "back")[(goto: (history:)'s last)]`.
4. Создайте блок footer с тегом footer. Это шаблон, который будет выводиться внизу страницы. Вставьте следующий код:`(if: (passage:)'s tags contains "inventory"))[](else: (link-repeat: "check inventory")[(goto: "inventory")])`

Вариант второй

1. Создайте отдельный параграф startup и добавьте ему тег startup. В блок вставьте следующий код:`(set: $inventory to (a:))`
2. Создайте блок footer с тэгом footer и вставьте следующий код: `<div class="inventory">(text: $inventory.join("\n"))</div>`
3. В левом нижнем углу откройте меню и выберите "Редактировать таблицу стилей" и вставьте:

```
.inventory {
```

```
    border-left: 1px #000 solid;
    float: right;
    top: 10%;
    left: 80%;
    position: fixed;
    padding: 20px;
    display:block;
}
```

Теперь ваш инвентарь отображается в блоке справа на каждой странице.

Кстати

Я использую объединенный вариант, потому что решил сделать систему с инвентарем и журналом.

[Рецепты: Сборка большого проекта]

При создании большого проекта, состоящего из over9000 страниц, или совместном написании проекта могут возникнуть трудности с размером файла. Поэтому предлагаю такой рецепт:

1. Создавайте историю как несколько блоков – в Twine это можно сделать создавая их как отдельные истории с именами Имя истории-1, Имя истории-2 и т.д.
1. Проверьте, что у ваших историй есть входы и выходы – они могут висеть пустыми. Главное чтобы окончание истории имело продолжение в другой, с таким же названием. После чего можете удалить висячие параграфы
2. Проверьте, не повторяются ли у вас общие блоки. Это могут быть характеристики персонажа, инвентарь, журнал. Удалите эти блоки из всех историй, кроме, например, первой или последней.

3. Создайте пустой проект с названием Имя истории и удалите из него автоматически созданный блок.
4. Проверьте любую мелочь, которая может отвалиться.

2. Скачайте и установите **Notepad++**. Он переваривает крупные файлы и у него есть подсветка синтаксиса.
3. В верхнем меню выберите Twine -> Показать библиотеку. После чего Twine нужно закрыть, он нам понадобится в самом конце.
4. Выделите все файлы, относящиеся к истории, и откройте их в Notepad++.
5. Структура у проекта следующая:`<tw-storydata><style></style><script></script><tw-passagedata></tw-passagedata>...<tw-passagedata></tw-passagedata></tw-storydata>`.Перенесите содержимое блоков <script></script> и <style></style>, если у вас там что-то есть.
6. Пройдитесь по всем файлам истории, делая следующее: замените `pid="` на `pid="0n`, где n – номер истории, который вы редактируете. Это позволит избежать конфликтов в игре после ее слияния в один файл.Так же удаляйте все блоки <script></script> и <style></style>, потому что мы их уже перенесли.
7. Начинайте по очереди – из каждого файла – переносить содержимое между тэгов `<tw-storydata></tw-storydata>`. Переносить их нужно в общий файл, который мы создавали самым последним.
8. Когда содержимое всех частей истории перенесено, эти файлы можно закрыть. В начале общего файла найдите строку startnode="1" и замените значение на "011", чтобы не потерять старт истории.

9. Сохраните изменения в общем файле и закройте его. Откройте Twine и подождите пока он прогрузит историю. И я не советую вам открывать ее в редакторе. Это будет очень медленно.

Когда Twine прогрузится, в правом верхнем углу истории (общий файл, без цифр), нажмите на иконку шестерни и выберите Опубликовать в файл. Остальные действия такие же, как были описаны выше, в разделе Публикация.

Полезные ссылочки

Собственно **Twine 2**. В правом верхнем углу нажмите Download и получите копию под свою OS.

Руководство по Harlowe.

All about Twine 2 and Harlowe – британский сайт с рецептами.

Моя собственная папка **Twine 2** в Google Drive, где я буду выкладывать наработки, статьи и переводы руководств. Но это не быстро – одно только руководство по Harlowe занимает 50 страниц.