Universität des Saarlandes, Wintersemester 2020/21
Ü:"Digitale Lernsettings - Narrativierung und game based learning"
Linda Hammann & Christine van Hoof

Twine-Spickzettel

Basierend auf Twine 2 und Harlowe 3.1.0 - erstellt für die #OERcamp Webtalks von Nele Hirsch (alles CC0 1.0), erweitert und bearbeitet von Linda Hammann für GUdig 2020 (alles CC0 1.0).

Erste Schritte

Auf www.twinery.org/2 gehen:

Anlegen einer neuen Geschichte	Im Anfangsmenü: “+Geschichte”
Remixen einer bestehenden Geschichte	Im Anfangsmenü: “Aus Datei exportieren”
Datei speichern **Ganz wichtig! Sonst geht’s verloren, wenn ihr euren Cache löscht!**	Im Storymenü, neben dem Titel eurer Geschichte: “Als Datei veröffentlichen” → erstellt eine html-Datei, die ihr lokal auf eurem Computer speichern und andern schicken könnt.

Grundlagen

Erstellung eines neuen Pfads mit doppelten, eckigen Klammern: *Ohne Benennung:* Er ging in [[die Schule]] *Mit Benennung (empfohlen)::* *[[Link-Text->Name der Passage]]* *oder* *[[Name der Passage <- Link-Text]]*	Er ging in die Schule
Benennung einer Passage:: mit einem -> hinter dem Text, der dem User angezeigt wird, z.B. [[die Schule-> Schule]] Eine gute Benennung ist wichtig, um den Überblick zu behalten und schnell von einer auf die andere Passage verlinken zu können. Indem ihr dann bei einem neuen Pfad den	*Im Erstellmodus sieht das dann so aus:* Er ging in [[die Schule ->Kap. 1.001]] *Im Spielmodus ist die Angabe unsichtbar:* Er ging in die Schule Der Link führt aber in die Passage namens Kap. 1.001.

Namen einer bestehenden Passage eingebt, verlinkt ihr auf diese zurück. Falls die Passage noch nicht existiert, erstellt Twine sie automatisch.	
Umbenennung einer Passage	Wollt ihr eine Passage umbenennen, macht ihr das in der Passage selbst, ganz oben. Dies ändert auch in allen Verlinkungen den Namen automatisch.
//kursiv//	*kursiv*
`''fett''`	**fett**
~~durchgestrichen~~	~~durchgestrichen~~
^^hochgestellt^^	hochgestellt
`Sonderzeichen, die nicht verändert werden sollen`, z.B. `&`	&
* Aufzählung	• Aufzählung
Externe Verlinkung, z.B. auf Wikipedia	 Linktext z.B. Linktext
Bild	<img src=”URL des Bildes”> (Upload z.B. auf imgur.com oder Verlinkung auf bestehendes Bild, z.B. in Wikimediacommons)
Gif	<iframe src="URL des Gif" </iframe>
Audio	<audio src="URL der Audio**datei**" autoplay loop>
Video	<video src="URL des Videos" controls height="400px"> </video>

Die Größe der Anzeige eurer integrierten Medien reguliert ihr mit Attributen, die ihr hinter die URL einfügt, z.B.

height="*Pixelangabe*px", z.B. height="300px"	Höhe der Anzeige
width="*Pixelangabe*px", z.B width="200px"	Breite der Anzeige
autoplay	Video startet automatisch, sobald Seite geöffnet wird
loop	Video läuft in Endlosschleife
controls	Video wird mit Kontrollpanel ausgestattet (empfohlen)

Ein Bild als Passagenlink setzen

Beim Klick auf das Bild kommt die Spieler zur nächsten Passage. Dazu einfach den img-tag in den Passagen-Link integrieren:

[[<img src="URL der Bilddatei"> ->Name der nächsten Passage]]

Gemeinsame Arbeit an einem Twine

Wichtig: Arbeitet mit deutlich unterscheidbaren Kapitel- bzw. Passagenamen und Variablen! Sonst kommt es zu Fehlern beim Zusammenfügen! Styles und Formatierungen sollten erst erledigt werden, wenn die Story in einer einzigen Datei gebündelt ist.

Möglichkeit 1): Asynchrone Zusammenarbeit
→ Person A arbeitet am Twine, lädt die Datei runter und schickt sie an Person B. Jetzt arbeitet Person B daran weiter, lädt die Datei runter, schickt sie an Person C usw.

Möglichkeit 2): Synchrone Zusammenarbeit
→ Notwendige Bedingung: Klares Storyboard, klarer Plot, alle Aspekte der Story samt Aufgaben sind festgelegt.
Die Story wird in Kapitel geteilt und diese klar aufgeteilt, z.B. Person A schreibt Kapitel 1, Person B Kapitel 2 usw.
Jedes Teammitglied erstellt seinen Twine-Teil mit eindeutiger und einzigartiger Benennung. Diese Einzelteile werden danach zusammengefügt und die Kapitel so miteinander verlinkt, dass eine kohärente Story entsteht.

Merging von Twine-Storys, d.h. Storyteil 1 soll mit Storyteil 2 verbunden werden:

- Beide Dateien werden runtergeladen und gleichzeitig in einem HTML-Editor geöffnet.
- in Datei “Storyteil 1”: suche den Tag <tw-storydata> und den dazugehörigen Endtag </tw-storydata>. Dafür muss man teilweise lange nach unten scrollen. Zwischen diesen Tags befinden sich alle Infos zu den Passagen der Story von “Storyteil 1”. Und dort, nach dem Ende der Passage-Data von “Storyteil 1”, also vor dem Endtag </tw-storydata>, wird gleich die gesamte Passage-Data der Datei “Storyteil 2” eingefügt. Für die bessere Lesbarkeit des Codes fügst du dort am besten noch einen Kommentar ein, z.B:

  ```
  <!-- Hier beginnt Storyteil 2 -->
  ```

- in Datei “Storyteil 2”: suche ebenfalls den Tag <tw-storydata> und den dazugehörigen Endtag </tw-storydata>, zwischen denen sich alle benötigten Passage-Data der Datei “Storyteil 2” befinden. Alle Infos zu den einzelnen Passagen beginnen mit <tw-passagedata> und enden mit </tw-passagedata>. Wir benötigen alle einzelnen <tw-passagedata>-Zeilen aus “Storyteil 2”. Diese werden mit Copy & Paste in die Datei “Storyteil 1” unter dem Kommentar, aber noch vor dem Endtag </tw-storydata> eingefügt.
- dann wird der interne Name der fusionierten Story geändert: Beim Tag <tw-storydata> ändert ihr das Attribut name=”Storyteil 1” zu name=”Storyteil 1+2”. Jetzt wird die neue, erweiterte Datei unter einem neuen Namen “Storyteil 1+2” gespeichert, damit nicht aus Versehen “Storyteil 1” überschrieben wird und dann verloren geht. Zum Testen wird die neue Datei “Storyteil 1+2” dann in Twine importiert und geöffnet.
- wahrscheinlich überlagern sich nun die Storys, d.h. die Passagen liegen übereinander, aber das lässt sich mit der Drag-Funktion leicht auseinandernehmen.
  → um Überlagerung direkt im Code zu verhindern: Position der Passagen von “Storyteil 2” verändern, z.B. + 500 bei x-Koordinate dazu nehmen, das verschiebt alles nach rechts.
  → Weitere Möglichkeit: Erstellt zu Beginn eine einzelne Datei und legt direkt alle Anfangspassagen der Kapitel großzügig nebeneinander. Arbeitet dann nur von oben nach unten beim Erstellen neuer Passagen, nicht von links nach rechts. Dann müsste beim Fusionieren alles ohnehin nebeneinander auftauchen.

Aussehen/ Design des Twine

Um dem Twine ein anderes Aussehen zu geben, wählt ihr im Storymenü unten links ‘Stylesheet der Geschichte bearbeiten’. In einem neuen Twine ist noch nichts eingetragen, da per Default ein schwarzer Hintergrund und weißer Text bzw. violetter Text für die Links eingestellt ist. Wollt ihr das ändern, könnt ihr mit CSS-Befehlen die Defaulteinstellung überschreiben.
Folgendes ist ein Basis-Design, das Du nutzen und auch beliebig anpassen kannst:

```
html {
  font-family: helvetica, arial, sans-serif;
  background-color:  #396fc6;
    color: black;
}
tw-story {
    width: 100%;
    font-size: 2em;
    line-height: 2em;
  background-color: white;
  color: #13102d;
 }


tw-link, .enchantment-link {
    color: #396fc6;
    text-decoration: none;
}
```

Mehr Farbcodes gibt's hier: https://developer.mozilla.org/de/docs/Web/CSS/Farben

Um den Überblick nicht zu verlieren, kann es nützlich sein, Kommentare in euer Stylesheet zu integrieren. Diese werden nicht als Code gelesen und sind daher nicht im Endergebnis sichtbar. Kommentare beginnen mit /* und enden mit */.

```
/*Das ist ein Kommentar im Stylesheet, der keine Auswirkungen auf den Look des Spiels hat.*/
```

Vor- & Zurückbutton deaktivieren

Ihr wollt nicht, dass die User in der Geschichte vor und zurückspringen und somit Entscheidungen rückgängig machen können? Dann entfernt den entsprechenden Button im Stylesheet der Geschichte fügt ihr dazu folgenden Code ein:

```
tw-sidebar {
display:none
}
```

Zwei Beispiel-Stylesheets zum verändern und anpassen gibt's hier. Einfach in euer Twine-Stylesheet reinkopieren und rumprobieren.

- Einfacher "Zauberwald"-Style: https://ebildungslabor.github.io/twine-webtalk//zauberwald.css

- Fortgeschrittene Übersicht mit vielen Features: https://ebildungslabor.github.io/twine-webtalk/ueberblick.css

Noch mehr Möglichkeiten

Makros & Variablen

Mit Makros (auch Funktionen genannt) lässt sich eine Twine-Geschichte weiter verfeinern. Makros sind Folgen von Anweisungen oder Deklarationen, die anstelle von Einzelanweisungen an mehreren Stellen im Programm mit nur einem einfachen Aufruf ausgeführt werden. Alle Anweisungen des Makros werden automatisch an der Programmstelle ausgeführt, an denen das Makro codiert wurde. Makros werden direkt im Erstellungstext der Passagen eingegeben. Wollt ihr viele Variablen benutzen, bietet es sich an, dafür eine gesonderte Passage noch vor dem Start der Geschichte anzulegen, in der ihr die Ausgangswerte der Variablen festlegt.

Variablen

Variablen haben im Spiel einen bestimmten, veränderbaren Wert, z.B. HP, Punktzahl, Geld, Zeit, etc.
Ressourcen als Variablen starten oft bei 0 und erhöhen bzw. verringern sich im Verlauf des Spiels und durch die Entscheidungen des Spielers.

Typen in Twine:

a. Boolesche Variable, auch logische Variable: Variable, die nur einen von zwei Werten haben kann: Wahr oder Falsch, also 1 oder 0, wie im Binärcode Ein/Aus. In Twine *true* oder *false*.
b. Integer-Variable: Form einer Variable, die nur Zahlen ohne Nachkommastellen, also Ganzzahlen, als Wert haben kann, z.B. Anzahl von Goldstücken: 100.
c. Zeichenkette: Wert ist ein Text, z.B. für Namen des Spielers. Der Wert von Zeichenketten muss immer zwischen Anführungszeichen stehen!

$ leitet eine Variable in Twine ein, z.B. $Spielername. Der Name der Variable kann alles sein, wichtig ist nur, dass er mit $ beginnt! Wählt eindeutige Namen, damit ihr immer wisst, worauf sich die Variable bezieht.
Eine Variable wird ausgelöst durch eine Funktion, also Code, der Twine eine bestimmte Aktion ausführen lässt, wenn die Variable in der Geschichte auftaucht.
Eine Funktion wird folgendermaßen in Twine eingegeben und der Code direkt in den Baustein kopiert:

(set: $*NamederVariable* = *Wert*)	Integer-Variable:

	(set: $Minutenzuspät = 0) Zeichenkette: (set: $Spielername = ”Anton”) Boolesche Variable (set: $SpielerhatSchlüssel to false) oder (set: $SpielerhatSchlüssel to true)

Beim Erstellen des Storytextes fügt ihr nun jedes Mal, wenn die Variable angezeigt werden soll, im Editor $*NamederVariable* ein. Twine zeigt dann automatisch den festgelegten Wert.

Übersicht (nicht vollständig)

$	führt eine Variable ein, z.B. einen individuellen Namen: $name
(set: $*Variable* = *Wert*) oder (set: $*Variable* to *Wert*)	Legt den Wert der Variable fest
(set: $*Variable* += *Wert*)	Erhöht den Wert der Variable.
(set: $*Variable* -= *Wert*)	Verringert den Wert der Variable.
(set: $*Variable* to false)	Setzt den Wert auf falsch.
(set: $*Variable* to true)	Setzt den Wert auf richtig.
(if:) *und* (then:)	Legt Bedingungen fest, Erklärung siehe unten.
(text-style:)	Legt einen Texteffekt fest, s.u.
(display: “Name der Passage”)	Zeigt den Text in der entsprechenden Passage an. Nützlich, wenn der gleiche Text an mehreren Stellen der Geschichte auftaucht.

Personalisierung des Spiels

Mit einer Zeichenkette lässt sich der Protagonist der Geschichte, den der Lernende spielt, mit einem Namen versehen.

```
(set: $Spielername = ”Anton”)
```

Alternativ kann eine Auswahl vorgegeben und daraus gewählt werden. Als Default ist im Beispiel unten “Max” eingestellt, durch Klicken auf den Link ändert der Spieler den eingestellten Namen.

```
(cycling-link: bind $name, 'Max', 'Maria', 'Markus', 'Mascha', 'Mara')
```

Soll der Spieler einen eigenen Namen eingeben, muss ein eigener Abschnitt mit einer Namensabfrage integriert werden. Es erscheint dann ein Textfeld, in das der Spieler einen Namen eingeben kann. Der Code für die individuelle Namensabfrage lautet:

```
(set: $name to (prompt: "Wie lautet dein Name?", ""))
```

Immer wenn danach im Text $name auftaucht, wird der zu Beginn vom User ausgewählte Name automatisch eingefügt.

Wenn-Dann-Bedingungen

Bedingungen funktionieren in Twine folgendermaßen: Es wird abgefragt, ob eine bestimmte Bedingung erfüllt ist. Ist die Antwort ja, wird eine Aktion ausgeführt. Bei einer Verneinung bleibt diese Aktion aus und es geht “normal” weiter. Eine Verneinung kann auch eine alternative Aktion auslösen.
Bsp: Einkauf: Der Spieler will ein Item für 100 Goldstücke kaufen. Im Hintergrund wird die Frage gestellt “Hat der Spieler 100 Goldstücke oder mehr?”
→ wenn: Goldstücke des Spielers > 100, dann: Kauf des Items möglich
→ wenn: Goldstücke des Spielers < 100, dann: Hinweis “Nicht genügend Goldstücke”

Wenn-dann-Aussagen in Twine einbauen:

1. Variable einführen

Boolesche Variablen	Integer-Variable
(set: $SpielerhatSchlüssel to false)	(set: $Goldstücke = 100) *Im Lauf des Spiels ändert sich der Wert durch die Entscheidungen des Spielers, z.B.*

	(set: $Goldstücke +=5) → *Spieler bekommt 5 Goldstücke.* (set: $Goldstücke -= 20) → *Spieler verliert 20 Goldstücke.*

2. Bedingung festlegen

(if: $SpielerhatSchlüssel is true) [Sperr die Tür auf.]	(if: $Goldstücke is > 110) [Kauf den Edelstein.]

Oder-Angaben

Für Oder-Angaben gibt es sowohl das (else-if:) als auch das (else:)-Makro.
Beide sind Varianten des (if:)-Makros und ermöglichen zu programmieren, dass eine Bedingung nicht erfüllt ist - und eine entsprechende Konsequenz angezeigt wird.
(else-if:) nutzt man, um eine Alternative zu einer vorangegangenen Bedingung zu geben. Ist die vorangegangenen Bedingung (if:) erfüllt (true), wird die (else-if:)-Angabe nicht angezeigt. Ist die Bedingung nicht erfüllt (false), wird stattdessen die (else-if:)-Angabe angezeigt.
(else:) wird genutzt, wenn keine Angabe zur Variable in der Bedingung vorliegt.

Das obere Beispiel können wir z.B. ergänzen durch:

(else:) [Such weiter nach dem Schlüssel.]	(else-if: $Goldstücke is < 110 and > 90) [Der Edelstein ist zu teuer, aber du kaufst stattdessen eine alte Karte.] (else:) Du hast leider zu wenig Gold um etwas zu kaufen.

Multiple Enden einbauen

Mit Makros lassen sich auch mehrere Enden einbauen, die der Spieler je nach seinen Entscheidung “freischaltet”.
Zunächst muss dafür eine Variable eingeführt und auf 0 gesetzt werden.

```
(set: $Entscheidung to 0)
```

Die Passage, in der die ausschlaggebende Entscheidung getroffen wird, verändert dann die Variable entsprechend.

```
(set: $Entscheidung to 1)

Oder

(set: $Entscheidung to 2)

Oder

(set: $Entscheidung to 3)
```

Die Story kann dann nach belieben weitergehen, bis der Spieler zum Ende kommt.
In der Endpassage fügt man dann die entsprechende Bedinung ein. Twine merkt sich, welchen Wert der Spieler der Variable $Entscheidung im Spielverlauf gegeben hat und zeigt ihm dann das entsprechende Ende.

```
(if: $Entscheidung is 1)[
Erstes Ende.
]
(else-if: $Entscheidung is 2)[
Zweites Ende.
]
(else-if: $Entscheidung is 3)[
Drittes Ende.
]
```

Randomisierung

Mit dem Makro (random:) können Angaben zufällig gemacht werden, z.B. beim Würfelspiel.
Soll jedes Mal ein beliebiges Ergebnis angezeigt werden, könnt ihr das so programmieren:

```
(set: $Würfel to (random: 1,6))
```

An das Ergebnis könnt ihr dann mit (if:) auch eine Bedingung knüpfen.

Textdarstellung bzw. -effekte

Text kann in Twine vielfältig dargestellt werden.
Einige Beispiele:

(text-style: "shadow")[Beispieltext]	Text wird mit Schatten hinterlegt.
(text-style: "mark")[Beispieltext]	Text wird hinterlegt, wie bei einem Textmarker.
(text-style: "outline")[Beispieltext]	Textfüllung ist leer, nur Textrand wird angezeigt

(text-style: "blur")[Beispieltext]	Text verschwommen dargestellt.
(text-style: "blink")[Beispieltext]	Text blinkt.
(t8n: "pulse")[Beispieltext]	Text pulsiert beim ersten Erscheinen. Einmaliger Effekt.
(t8n: "flicker")[Beispieltext]	Text flackert mit kleiner Verzögerung auf. Einmaliger Effekt.
(t8n: "dissolve")[Beispieltext]	Text faded ein.
(t8n: "rumble")[Beispieltext]	Text poltert beim Erscheinen. Einmaliger Effekt.
(t8n: "shudder")[Beispieltext]	Text wackelt beim Erscheinen. Einmaliger Effekt.
(t8n: "slide-right")[Beispieltext]	Text fliegt von links nach rechts ein. Einmaliger Effekt. Funktioniert entsprechend auch für links.

Mehr Effekte: https://twinery.org/wiki/harlowe:transition-depart?redirect=1

Besondere Schriftarten nutzen

Soll im Twine eine besondere Schriftart benutzt werden, die der User möglicherweise nicht bereits installiert hat, habt ihr 2 Möglichkeiten. Frei verfügbare Fonts gibt's z.B. hier: www.fonts.google.com.

1) Verlinken:
   Ganz oben in euer Stylesheet tragt ihr folgenden Code ein:

```
@import url('URL der gewählten Schriftart')
```

2) @font-face: Sucht und wählt die Schriftart (samt evt Spezifikationen) aus. Kopiert die URL aus dem Embed-Link und fügt sie in euren Browser ein. Es öffnet sich eine Seite mit zahlreichen CSS-Befehlen, sogenannten @font-face-Regeln. Ihr braucht die Befehle, die als /*latin*/ gekennzeichnet sind. Manchmal sind diese Befehle in verschiedene Blöcke unterteilt, ihr braucht aber alle. Kopiert **alle** /*latin/-Befehlsblöcke und fügt sie ganz oben im Stylesheet der Twinestory ein.

Dann müsst ihr nur noch im Stylesheet angeben, wo die Schriftart benutzt werden soll, z.B.

```
@import url('URL der gewählten Schriftart')

html {
  font-family: 'Name der gewählten Schriftart';
  background-color:  black;
  color: white;
```

```
}
```

Übergänge

Die Übergänge zur nächsten Folie bzw. Passage können vielfältig gestaltet werden, z.B. 'zitternd'. Beispiel: (t8n-depart: "rumble")[[Name der Passage]]

Choreographie

Inhalte können nach bestimmten Regeln angezeigt werden:
(event: when time > 3s)[Text erscheint nach 3 Sekunden]
(link:"Klick")[Erst bei einem weiteren Klick erscheint weiterer Text]
(more:)[Text erscheint, wenn keine anderen Links mehr da sind.]

Weitere Befehle, mehr Makros und ausführliche Erläuterungen gibt es in der offiziellen Harlowe-Dokumentation (englischsprachig):
https://twine2.neocities.org/

Zusatzinfos:

Twine als komplette Offline-Version erstellen

Um unabhängig von einer bestehenden Internetverbindung zu sein (wenn z.B. das Schul-Wlan zu schlecht ist...), müssen alle Dateien, die im Twine verarbeitet sind (Bilder, Audio, Video, Schriftarten, etc.) lokal gespeichert werden. Leider können allerdings dann bestimmte Features, z.B. ein Kollaborationsraum, interaktive H5P-Inhalte oder die Verlinkung auf externe Webseiten zur weiteren Recherche nicht eingebaut werden. Außerdem kann das Twine dann zwar im Editor (online oder offline) erstellt, aber nicht getestet oder gespielt werden. Dies ist nur lokal möglich, nachdem die Twine-Datei heruntergeladen wurde.
Alle Dateien, die in der Story verbastelt sind, müssen in demselben Ordner gespeichert werden, in dem auch die html-Datei des Twines nach dem Download liegt, bzw. in einem Unterordner, z.B. “media”. Bei der Erstellung eines Offline-Twines werden nämlich keine externen, sondern relativen Links/Pfade zu den Dateien gesetzt, die lokal gespeichert sind.

Ist das Twine im Editor fertig erstellt, muss die Twine-html-Datei heruntergeladen, im Ordner, in dem auch die Dateien liegen, gespeichert und lokal im Browser ausgeführt werden. Erst dann sind die eingearbeiteten Dateien sichtbar.
Der gesamte Ordner kann dann Dritten zur Verfügung gestellt werden. Es wird empfohlen, dort auch direkt ein ReadMe mit einer Anleitung für das Twine sowie die Datei mit den Literatur- und Lizenzangaben zu hinterlegen.

Relative Pfade:

Bild	<img src="./media/NameDesBildes.jpg">
Audio im Hintergrund	<audio src="./media/NameDerDatei.endung" autoplay>
Steuerbare Audiodatei	<iframe src=”./media/NameDerDatei.endung” controls </iframe>
Video	<video src="./media/NameDerDatei.endung" controls autoplay> </video>

Einbindung einer besonderen Schriftart: Dafür muss die Fonts-Datei zunächst heruntergeladen und im gleichen Ordner wie alle anderen Medien gespeichert werden (z.B. “media”). Im nächsten Schritt wird die Schriftart installiert. Dann öffnet ihr das Stylesheet eurer Geschichte und fügt ganz oben Folgendes ein:

```
@font-face { font-family: ‘Name der Schriftart’;
        src: local (‘Name der Schriftart’)
            url('./media/Dateibezeichnung.ttf') format('truetype'); }
```

Zuletzt müsst ihr nur noch angeben, wo die Schriftart benutzt werden soll. Dann sieht euer Stylesheet z.B. so aus:

```
@font-face { font-family: ‘Name der Schriftart’;
        src: local (‘Name der Schriftart’)
            url('./media/Dateibezeichnung.ttf') format('truetype'); }

html {
  font-family: ‘Name der Schriftart’;
  background-color:  black;
    color: white;
}
```

Wie bei allen Medien im offline-Modus wird die Schriftart aber im Spiel erst sichtbar, wenn ihr die Datei exportiert und lokal im Browser startet. Der Editor findet den lokalen Pfad nicht. Ihr müsst dann zudem eure User darauf aufmerksam machen, dass sie vorm Starten des Twines ebenfalls die Schriftart installieren müssen.