EndlessBook - User Manual

A guide to using the EndlessBook plugin by echo17.

Purpose

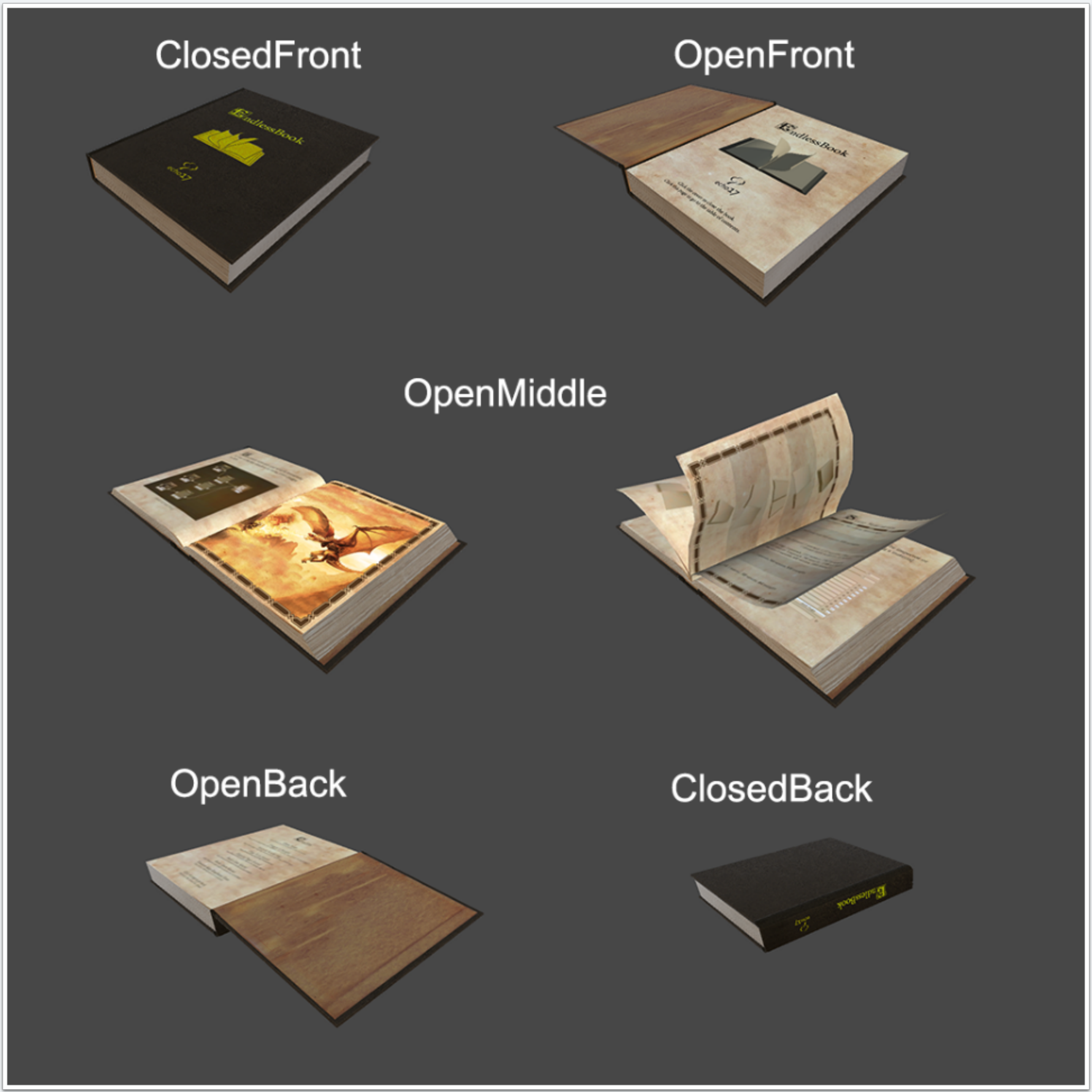
EndlessBook is a complete way to visualize a book inside your Unity scene. EndlessBook is more than just an animated mesh. It comes complete with scripts to handle updating your pages, animating the book states and page turns, swapping out static models when not animating, and firing handlers when the book changes state or starts / stops a page turn.

States of the Book

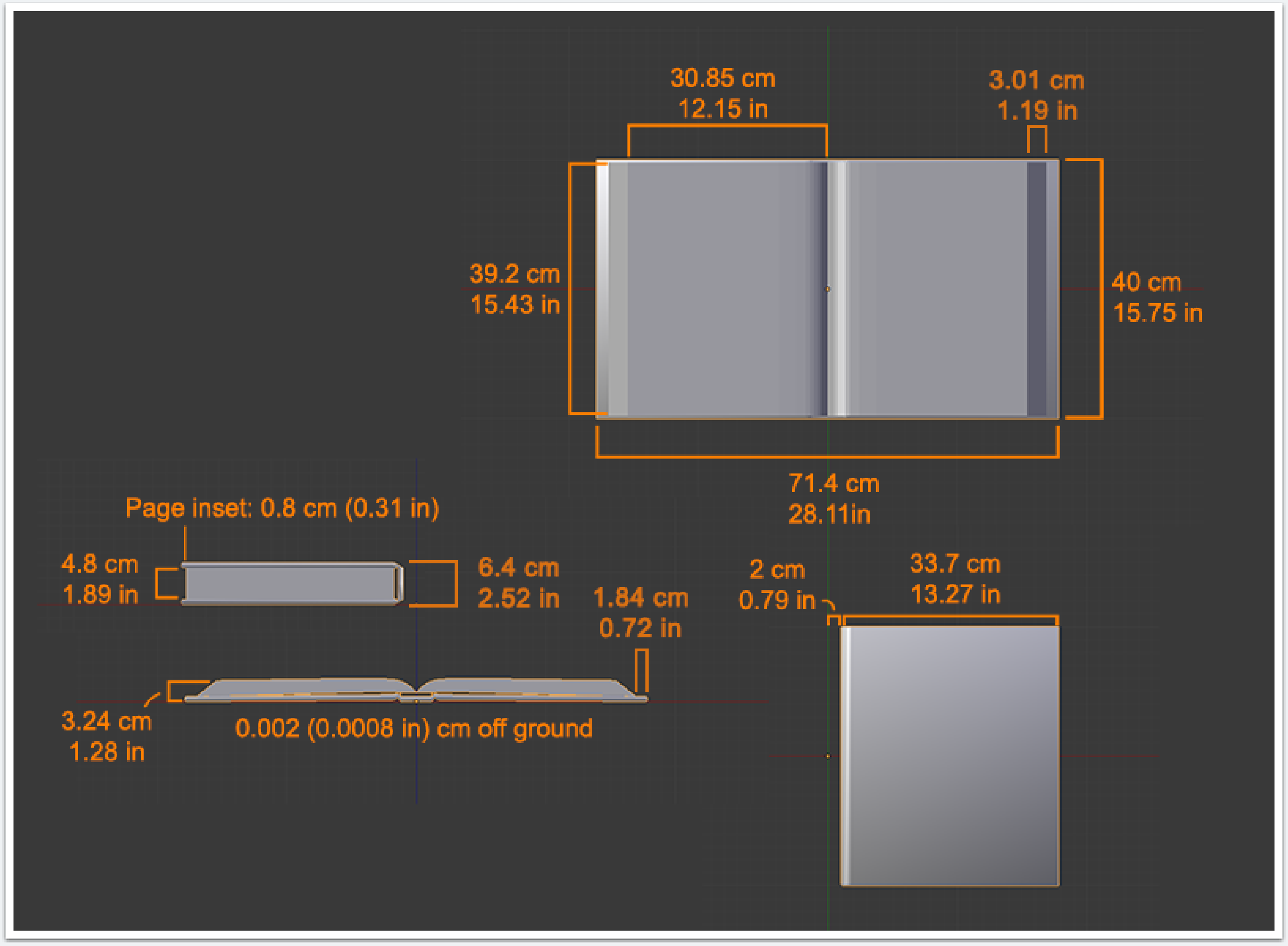
There are five states to EndlessBook

- ClosedFront
- OpenFront
- OpenMiddle
- OpenBack
- ClosedBack

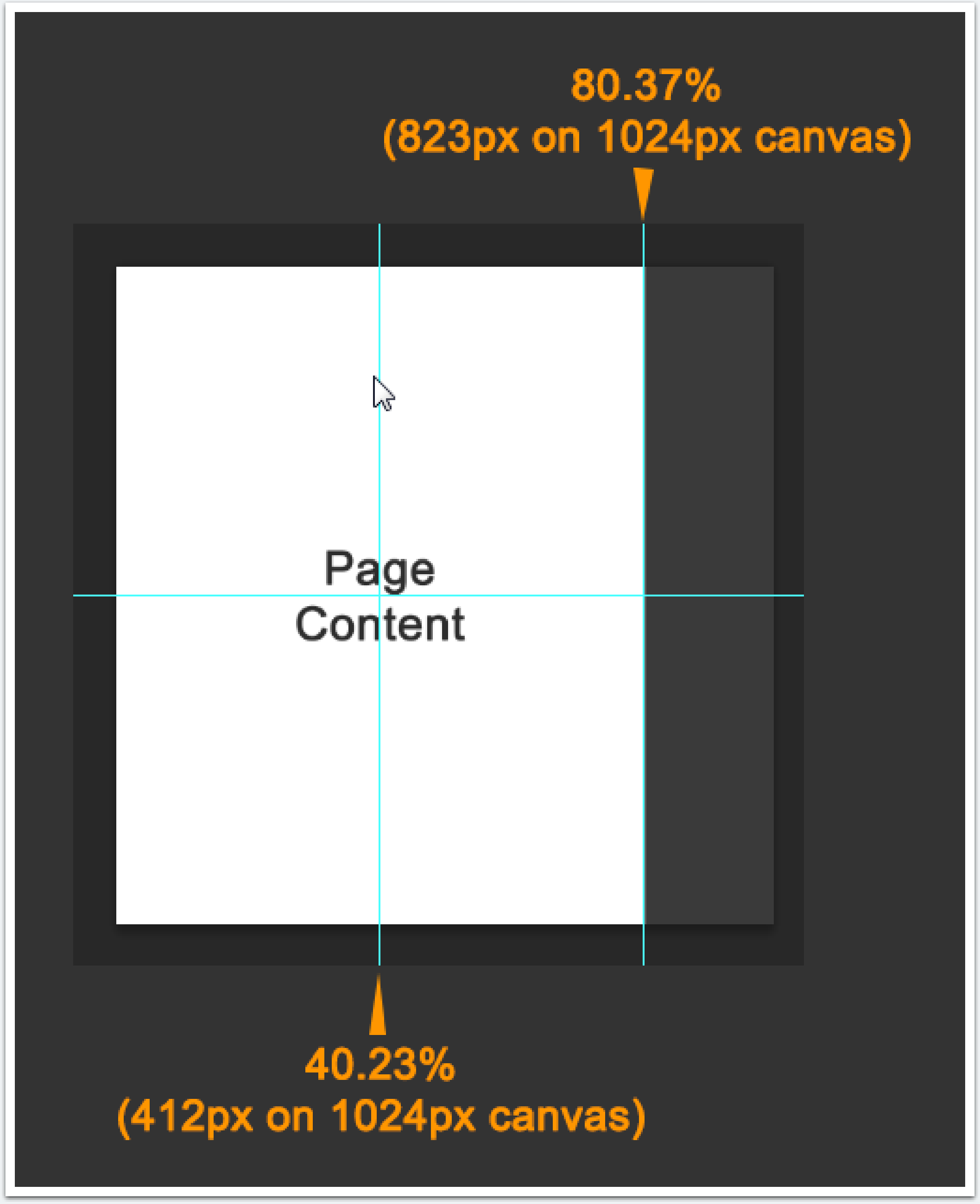
Pages are only turned when in the OpenMiddle state.



Book Mesh Dimensions

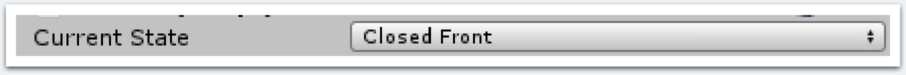


Page Texture Dimensions



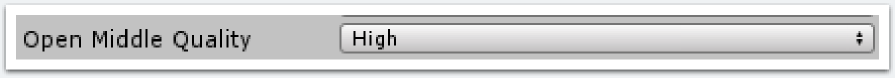
Book Inspector

To change the current state of the book from the inspector, simple choose the new state you would like. The book will immediately swap to that static mesh in the scene.



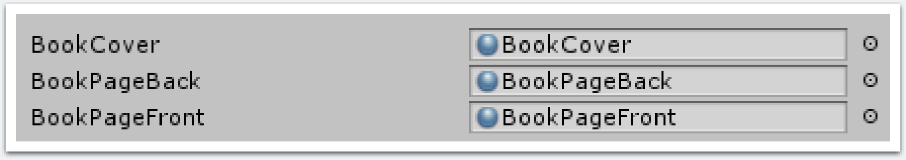
Current State: Closed Front

To set the quality of the OpenMiddle state, select from one of three quality levels (see Standins section).



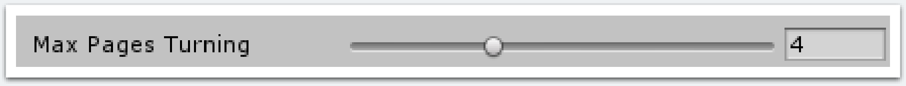
Open Middle Quality: High

Drag materials to set the BookCover, BookPageBack, and BookPageFront sections of the meshes. The BookCover wraps around the cover and page edges. The front and back pages are usually only visible when in the OpenFront or OpenBack states.

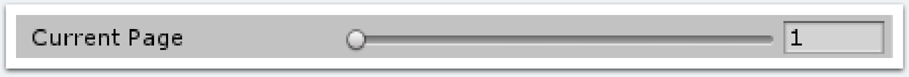


BookCover	BookCover
BookPageBack	BookPageBack
BookPageFront	BookPageFront

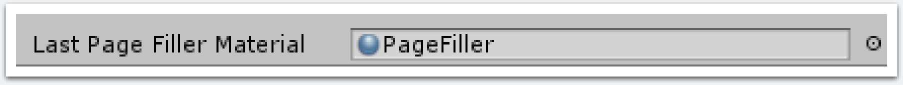
Set the Max Pages Turning value to change how many pieces of paper are allowed to turn when changing page numbers.Too few pages will make the page turn seem unaturally fast and spaced out. Too many will slow down the processor and bog the memory with lots of materials. Finding a balance here is something that needs to be done on a project-by-project basis. You can use the slider or set the value in the textbox.


Max Pages Turning 4

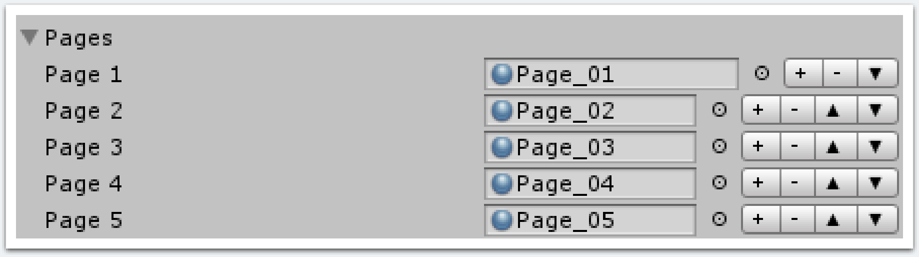
Current Page immediately jumps to the page you set. You can use the slider or set the value in the textbox.


Current Page 1

To set the last page filler that is used on the last page if your page count is uneven, drag a material to the Last Page Filler Material slot. This will also be used as the default for all new pages added.


Last Page Filler Material PageFiller

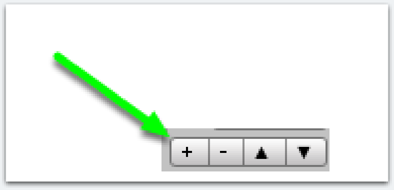
Drag materials to each of your pages to set what will be displayed on them.



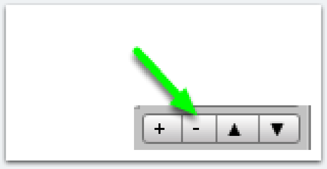
Pages

Page 1	Page_01
Page 2	Page_02
Page 3	Page_03
Page 4	Page_04
Page 5	Page_05

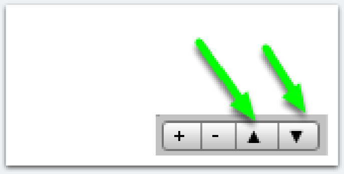
To insert a page between two other pages, click the + (plus) button. This will insert the page above this page.



To remove a page, click the - (minus) button.



To move a page up or down, click the appropriate arrows.



To add a page to the end of the list, click the Add Page button.

Controlling the Book

You can set most of your book's properties from the inspector of the Book component. For animations and interaction, you will need to use scripting.

There are several methods available in the Book script for you to control your book at runtime.

SetState

void SetState(StateEnum state, float animationTime = 1f, StateChangedDelegate onCompleted = null)

This will set the state of the book in the time specified by animationTime. When the book has completed the animation and changed to the new state, the onCompleted handler will be fired if it is not null.

Example:

```
book.SetState(Book.StateEnum.OpenMiddle, 0.7f, BookStateSet);
```

TurnForward

void TurnForward(float time, StateChangedDelegate onCompleted = null, PageTurnDelegate onPageTurnStart = null, PageTurnDelegate onPageTurnEnd = null)

This turns the book forward one page. When the turn is completed, onCompleted will fire if not null. When the page starts to turn, the onPageTurnStart handler will be fired if it is not null. When the page has completed turning, the onPageTurnEnd handler will be fired if it is not null. The onCompleted delegate is a state change delegate and has different parameters than the onPageTurnDelegate, but both fire in succession at the end of the page turn in this method.

You can use the delegates to activate and deactivate elements of your pages, like render texture cameras, or to play sounds.

Example:

```
book.TurnForward(0.8f);
```

TurnBackward

void TurnBackward(float time, StateChangedDelegate onCompleted = null, PageTurnDelegate onPageTurnStart = null, PageTurnDelegate onPageTurnEnd = null)

This turns the book backward one page. When the turn is completed, onCompleted will fire if not null. When the page starts to turn, the onPageTurnStart handler will be fired if it is not null. When the page has completed turning, the onPageTurnEnd handler will be fired if it is not null. The onCompleted delegate is a state change delegate and has different parameters than the onPageTurnDelegate, but both fire in succession at the end of the page turn in this method.

You can use the delegates to activate and deactivate elements of your pages, like render texture cameras, or to play sounds.

Example:

```
book.TurnBackward(1.1f, TurnCompleted, PageTurnStart, PageTurnEnd);
```

TurnToPage

void TurnToPage(int pageNumber, PageTurnTimeTypeEnum turnType, float time, float openTime = 1f, StateChangedDelegate onCompleted = null, PageTurnDelegate onPageTurnStart = null, PageTurnDelegate onPageTurnEnd = null)

This turns the book to a page, regardless of what the current page is. If the page number is greater than the current page, the book will turn forward, otherwise backward. The turnType parameter specifies how the time parameter should be used. If

TotalTurnTime is used, the time value will represent the total amount of time to turn all pages. If TimePerPage is used, the time value will represent the amount of time to turn each page individually regardless of how many total pages are turning.

If the book is not in the OpenMiddle state, the book will first transition to the OpenMiddle state in a duration of the openTime parameter.

When the book has completed turning to the page, the onCompleted handler will fire. When each page starts its turn, the onPageTurnStart handler will fire. When each page ends its turn, the onPageTurnEnd handler will fire.

You can use the delegates to activate and deactivate elements of your pages, like render texture cameras, or to play sounds.

Example:

```
book.TurnToPage(34, Book.PageTurnTypeEnum.TotalTurnTime, 10.1f);
```

Other Methods

void StopTurningPages(): Immediately stops the page turning animations and sets the page to the value that was specified in one of the page turn methods.

PageData GetPageData(int pageNumber): Returns the page data (material) for a given page number.

PageData AddPageData(): Creates a new page in the book and returns the data. You can use the return value to set the material.

PageData InsertPageData(int pageNumber): Creates a page in the book before the specified page number and returns the data. You can use the return value to set the material.

void SetPageData(int pageNumber, PageData data): Sets the material for a page at the page number.

void RemovePageData(int pageNumber): Removes a page from the book at the page number.

Methods that are mostly for the inspector to use at design time, but are included for completeness:

void SetPageNumber(int pageNumber): This will set the page number immediately, skipping the animations. This is the method called by the inspector when the Current Page is updated.

void SetStandinQuality(StateEnum state, StandinQualityEnum quality): Sets the static mesh standin quality for a given state (currently only OpenMiddle is supported). This is rarely used at runtime and will likely be set once in the inspector at design time from the inspector.

void SetMaterial(MaterialEnum materialType, Material material): Sets a material for the given type (cover, front page, back page). This would rarely change at runtime and should be set once at design time from the inspector. The BookPageLeft and BookPageRight material types will be set by the plugin automatically as the pages are turned, so these two types can be ignored. They are only included for internal use.

void SetMaxPagesTurningCount(int newCount): Sets the maximum number of animated pages that can be displayed at one time. Too few pages will make the page turn seem unaturally fast and spaced out. Too many will slow down the processor and bog the memory with lots of materials. Finding a balance here is something that needs to be done on a project-by-project basis. This method is rarely called at runtime and should be set once at design time from the inspector.

void SetLastPageFillerMaterial(Material material): Sets the page filler material which is used on the last page if the page count is uneven. This is rarely called at runtime and should be set once during design time from the inspector.

Delegates

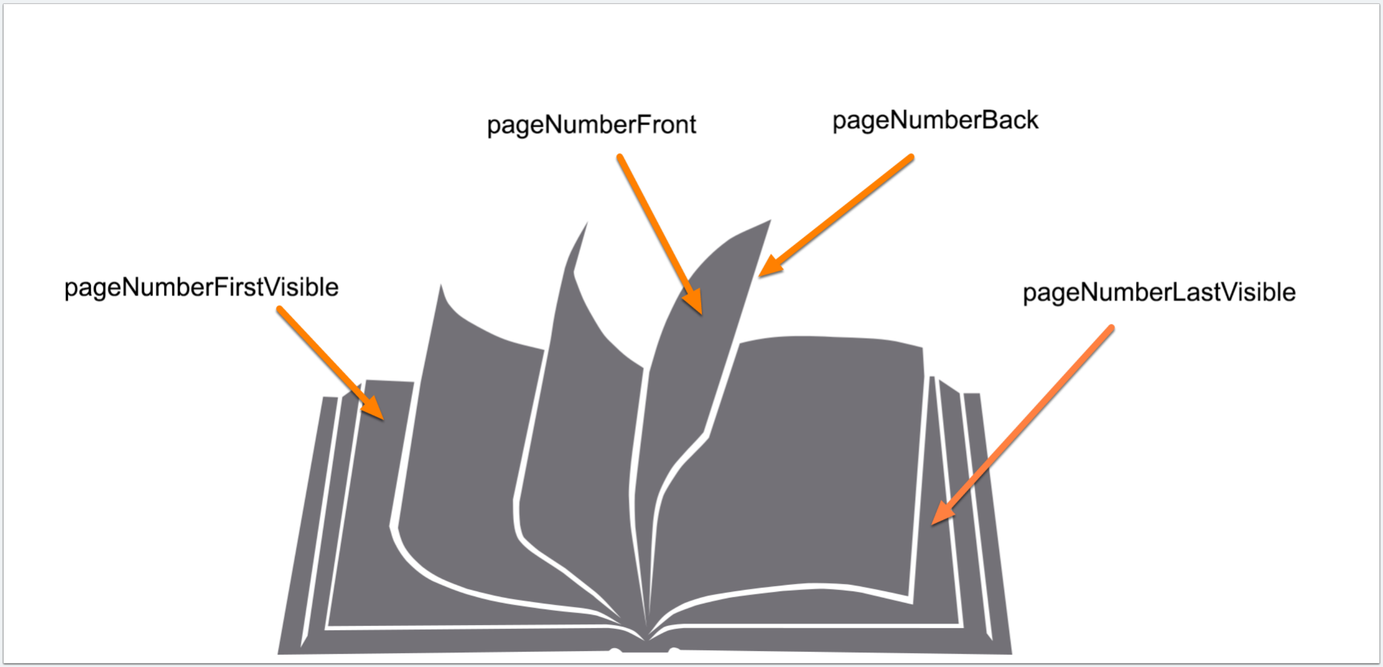
void StateChangedDelegate(Book.StateEnum fromState, Book.StateEnum toState, int pageNumber)

Used in the SetState, TurnForward, TurnBackward, and TurnToPage methods optionally.

void PageTurnDelegate(Page page, int pageNumberFront, int pageNumberBack, int pageNumberFirstVisible, int pageNumberLastVisible, Page.TurnDirectionEnum turnDirection)

Used in the TurnForward, TurnBackward, and TurnToPage methods optionally.

- pageNumberFront: the page number of the front of the turning page
- pageNumberBack: the page number of the back of the turning page
- pageNumberFirstVisible: the page number of the first visible page of the book at the moment the handler is fired
- pageNumberLastVisible: the page number of the last visible page of the book at the moment the handler is fired



Standins

EndlessBook tries to optimize performance. The animated book requires a lot of bones to give a natural bend to the pages, but that is a lot of overhead to process, especially since the book will likely be open or closed most of the time and not changing state. To balance having the realism of the bending paper with no overhead during downtime, EndlessBook will swap out the animated skinned mesh with a static mesh renderer and a standin model. Each of the five states has a standin, with the OpenMiddle state have three: High Quality, Medium Quality, and Low Quality.

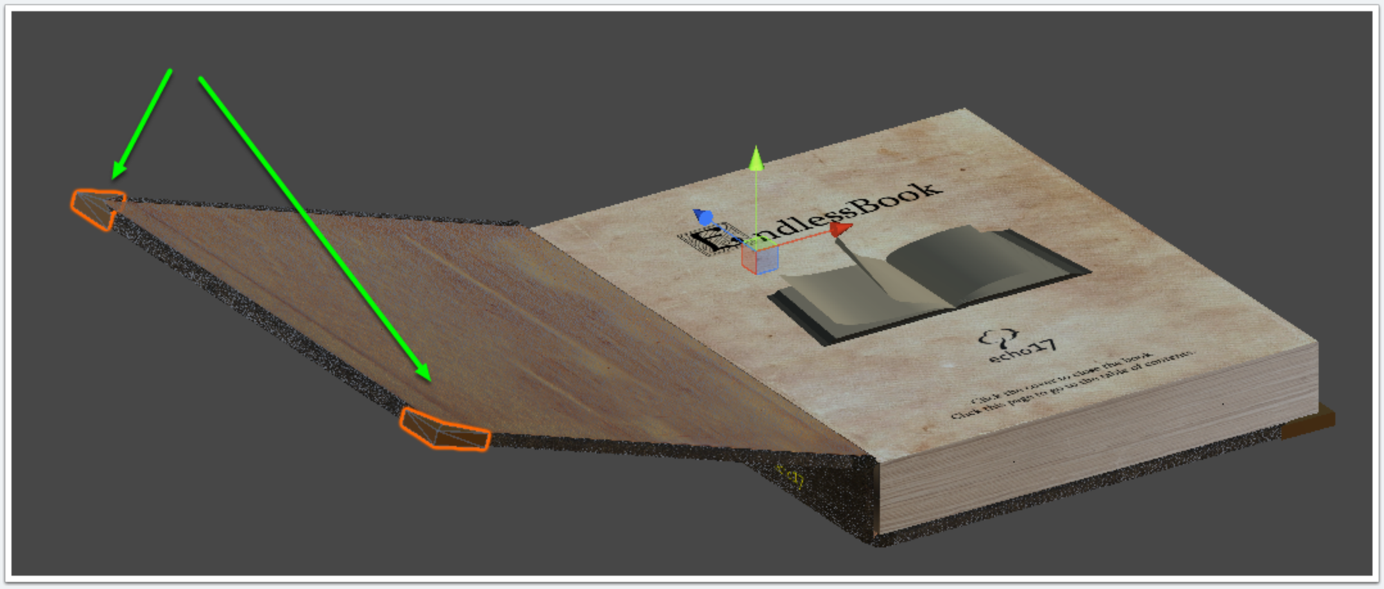
High Quality: Full geometry and materials of the animated model, minus the bones and skinned mesh renderer. Suitable for all angles since nothing is taken away.

Medium Quality: Most of the geometry and materials of the animated model, minus the bones, skinned mesh renderer, and the front and back pages. Most of the time the front and back pages will be hidden anyway, so this quality level will be suitable for almost all camera angles.

Low Quality: Minimum geometry and materials needed to show the book from an overhead, orthographic camera. This level has no front and back materials, no bones or skinned mesh renderer, and no geometry for the page edges at the top and bottom of the book. In addition, the cover geometery for both the back and front of the book is removed since it will not be visible from this specific type of camera.

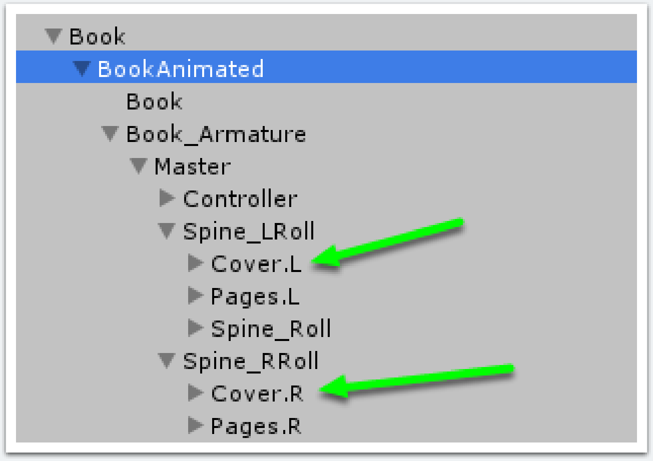
Optional Cover Customizations

EndlessBook is a simple book model with little frills to allow you to use it in any type of project. You can easily add elements to your book cover by creating the geometry in a modeling tool and then following the bone transforms of the book armature.



The front cover will need to follow: Book_Armature/Master/Spine_LRoll/Cover.L

The back cover will need to follow: Book_Armature/Master/Spine_RRoll/Cover.R



```
▼ Book
  ▼ BookAnimated
      Book
    ▼ Book_Armature
      ▼ Master
        ▶ Controller
        ▼ Spine_LRoll
          ▶ Cover.L
          ▶ Pages.L
          ▶ Spine_Roll
        ▼ Spine_RRoll
          ▶ Cover.R
          ▶ Pages.R
```

You can't simply parent to these bones since the animated mesh will only be visible and active while the book is changing state. Instead you will need to mimic your objects as being children of the bones. You can do that with a script included with Demo #2 called MimicChild.cs, setting the parent of each object to the bones listed above. This way your objects will be available and visible in all states, animated or not.

Render Textures

If you are rendering a sub-scene into a render texture to use as a page material, there are some things you can do to improve the performance of your game. Rather than having the render texture constanltly updated even when the page is not visible, you can use the PageTurnDelegates to tell your sub-scene to turn on and off. For example, when the onPageTurnStart handler is fired for a given page, you can turn your sub-scene on, thus rendering it to the newly visible page. When the page has completed turning, you can then turn the sub-scene off, saving processing power for something that is no longer visible.

Each animated page that turns has a front and back page, so you will need to take these into account. In addition, the book itself has first and last visible pages (left and right) that will adjust as the pages turn. For a complete example of how to handle this, please refer to the Demo #2 included with the plugin. The Demo02.cs script shows how you can manage your page view scenes as the book pages turn.

Interaction

How you interact with the EndlessBook is entirely up to you. Demo #2 comes with some ideas that might help you with your project, but they are by no means the only way to handle them.

Turning Pages

Demo #2 has a very simple touch pad collider for the left and right sides of the book. It captures where the user clicked and sends a message back to the demo to either change the book state or turn a page. You will likely want to have a more fleshed-out touch interface than this simple one for your project.

You certainly do not need to require a book to be touched to turn pages, however. You could have a page flip on a timer that has no interaction at all, for example.

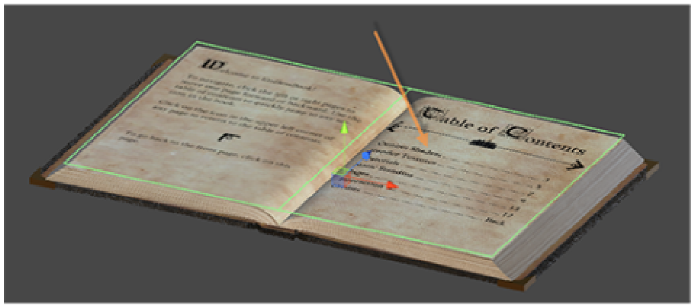
Interacting with a Render Texture Scene

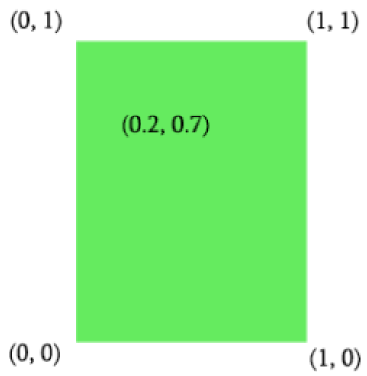
Demo #2 shows how you can interact with your render texture scenes.

The table of contents has several colliders in a scene. When the user touches the book's touch pad collider, it registers the hit point and normalizes it to the bounds of the touchpad collider (x axis goes from zero to one, y axis goes from zero to one). It then casts a ray into the table of contents scene from the scene's camera, seeing if it hits one of the chapter colliders. If so, it sends a message back to the demo which handles the page jump.

The demo book also contains a map interface that shows a very simple example of how you might use ray casting to scroll a map around, interacting with it by clicking on the continents.

Ray is cast into the main scene from the main camera and hits the touch pad collider. The hit point is normalized to the bounds of the touchpad collider.





Another ray is cast from the page scene camera using ViewportPointToRay(hitPointNormalized) to determine if any scene objects were touched.