renpy-cardgame Documentation

Release latest

May 29, 2015

Contents

Cardgame is a framework that provides primitives for creating cardgames with Ren'Py. The cardgame engine does not actually implement any card games for you. Instead, it implements a set of primitives that are common to various card games:

- Cards
- Stacks of cards
- Clicking and double-clicking on a card or stack.
- Dragging cards between stacks
- Flipping cards over

Providing these primitives makes it easy to implement card games. (For example, an implementation of klondike solitaire takes 225 lines of code, including comments and blank lines.)

CHAPTER 1

Downloading

Cardgame is now maintained on github at:

https://github.com/renpy/cardgame

It can be downloaded using git, or by clicking the “Download Zip” button at the bottom of the right column of that github link.

CHAPTER 2

License

Cardgame is licensed under the following terms, which are open source and free for commercial use.

```
Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation files
(the "Software"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge,
publish, distribute, sublicense, and/or sell copies of the Software,
and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

CHAPTER 3

Concepts

There are four concepts that are used by cardgame: tables, cards, stacks, and card events.

Table Table objects are the fundamental displayable used by cardgame. A table may have multiple stacks and multiple cards defined underneath it, and takes care of displaying cards and stacks, and producing events in response to user input.

Tables should be displayed on the master layer using the show and hide methods. Adding a table to the transient layer may lead to confused card motion animation.

Tables do not display a background image underneath the stacks. Such an image can be displayed using the scene or show statements.

Card Cardgame represents cards using hashable values provided by the user. Unlike tables, stacks, and events, cards are not objects produced by the cardgame, but instead can be objects or other values produced by the user. For example, the eight of spades might be represented as the tuple (1, 8). User-defined objects are also often acceptable.

Cards may be face-up or face-down. A card has two images associated with it: a face and a back. A card may either be part of a single stack, or may not belong to any stack. A card is only displayed if it belongs to a stack.

Stack A stack represents a group of one or more cards. Each stack has a position relative to the table, and cards are displayed at offsets relative to that position. Stacks support many of the methods of a list, and also a few cardgame-specific methods. Stacks have a base image that is shown when the stack is empty.

CardEvent When a table is shown and sensitive, CardEvent objects will be created in response to user input. These objects store information about the event.

CHAPTER 4

Table

class `Table`(*back=None*, *base=None*, *springback=0.1*, *can_drag=...*, *doubleclick=.33*)

Creates a new Table.

base A displayable giving an image used as the base of stacks that do not specify a more specific base.

springback The amount of time it takes for cards to spring to their new location when they have been moved between stacks, or dropped.

can_drag A function that should return true if a card can be dragged. The function is given three arguments, the table, stack, and card that are being dragged. It should return True to allow the drag, or False otherwise. The default function returns True if the card being dragged is face up.

doubleclick The maximum time between clicks for a doubleclick event to be recognized.

`card`(*value*, *face*, *back*)

Declares a new card.

value A hashable value that is used to represent this card.

face A displayable that's used for the face of this card.

back If not None, a displayable that's used for the back of the card. Otherwise, the value of the back argument to the Table constructor is used.

`stack`(*x*, *y*, *xoff=0*, *yoff=0*, *show=1024*, *base=None*, *click=False*, *drag=DRAG_NONE*, *drop=False*, *hidden=False*)

Declares a new stack of cards, and returns the Stack object.

x, y Give the x and y offsets of the center of the bottom-most card of the stack, or the base of the stack if no card exists.

xoff, yoff The offsets of each card in the stack relative to the next-lower card in the stack.

show The number of cards to show from the stack. If there are more than this number of cards in the stack, only the topmost show cards are shown.

base An image that is used for the base of the stack, if no cards are in the stack. If this is None, the default base specified with the Table is used.

clicked If True, this stack will return "click" and "doubleclick" events when the stack is clicked.

drag Sets which cards, if any, participate in drags from this stack:

- DRAG_NONE - No dragging can occur.
- DRAG_CARD - Only the card being dragged will be dragged.
- DRAG_ABOVE - The card being dragged and all cards above it will be dragged.

- DRAG_STACK - All cards in the stack will be dragged.
- DRAG_TOP - The top card in the stack will be dragged.

drop If True, this stack can be used as a drop target.

hidden If True, this stack will not be shown on the screen.

show(*layer='master'*)
Shows this table on *layer*.

hide(*layer='master'*)
Hides this table from *layer*

set_sensitive(*value*)
Determines if this table will respond to events. If value is false, the table will stop responding to events until this is called with *value* true.

set_faceup(*card*, *faceup=True*)
Determines if *card* will be displayed face up or face down. The card is displayed face up if *faceup* is True.

get_faceup(*card*)
Returns True if the *card* is faceup and False otherwise.

set_rotate(*card*, *rotation*)
Sets the rotation of *card* to *rotation* degrees. Rotation quality leaves something to be desired.

get_rotate(*card*)
Returns the rotation of *card*, in degrees.

add_marker(*card*, *marker*)
Adds a marker to the card. *marker* should be a Displayable.

remove_marker(card, marker):
Removes *marker* from *card*.

CHAPTER 5

Stack

class **Stack**

Stack objects represent stacks of cards, and are created by the Table.stack method.

Stack support basic list operations, like len(), indexing, membership tests, and iteration. Cards are placed in the list in bottom to top order.

Stack objects also support the following methods:

insert(*index*, *card*)

Inserts *card* in the stack at index, where 0 is the bottom of the stack and len(s) is the top of the stack. If the card is in a stack, animates the card moving to the new stack.

append(*card*)

Places *card* on the top of the stack. If the card is in a stack, animates the card moving to the top of the stack.

remove(*card*)

Removes card from the stack.

deal()

Removes the card at the top of the stack from the stack, and returns it. Returns None if the stack is empty.

shuffle()

Rearranges the cards in the stack in a random order.

CHAPTER 6

CardEvent

class `CardEvent`

CardEvent objects are returned from ui.interact() when events happen while a Table is sensitive. All event objects have the following fields defined:

type One of “drag”, “click”, or “doubleclick”, giving the type of event this is.

table The table this event is associated with.

stack The stack that has been clicked or dragged from.

card The card that has been clicked or dragged. None for a click on the stack.

Drag events also have the following fields defined.

drag_cards A list of cards being dragged.

drop_stack The stack the cards are being dropped on.

drop_card The card the cards are being dropped on, if any, If this is None, the cards were dropped onto an empty stack.

A

C

D

G

H

I

R

S

T