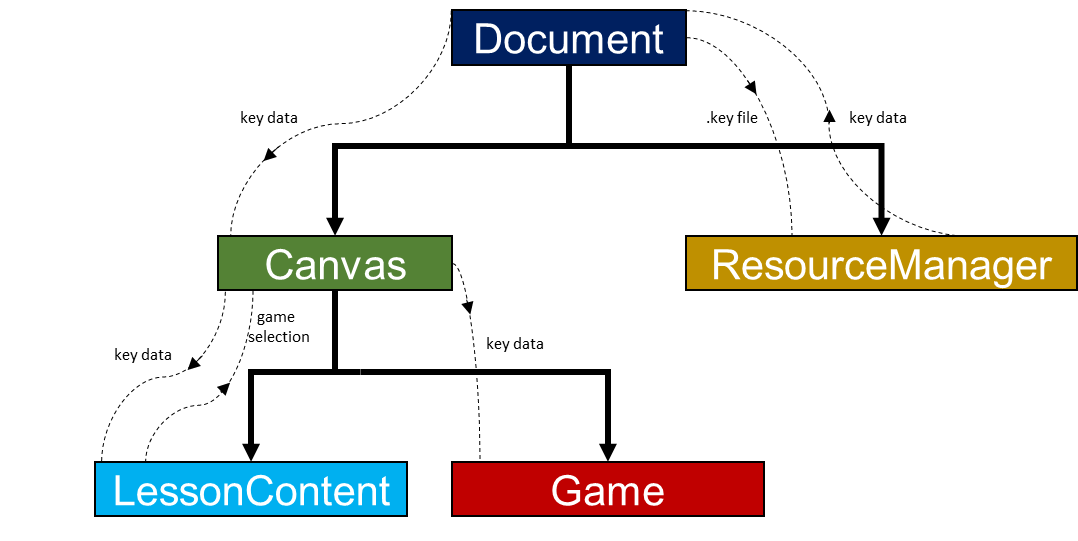
|  |  |  |
| --- | --- | --- |
| **Startup Process** | | |
| **Step** | **User/System Init.** | **Action** |
| 1 | System | *Document* listens for *InvokeEvent*, triggered on application start. |
| 2 | System | *Document* initializes *ResourceManager*. |
| 3 |  | *Document* listens for *ResourceManager* to complete loading a .key file. |
| 4 | System | If application is opened by an associated .key file, then the *onInvokeEvent* function will initialize *Canvas* with the file URL as an argument.\*  *Canvas* will open *LessonContent*.  *onInvokeEvent* will also send the file URL to *ResourceManager*. |
| 5 | System | If the application is not opened by an associated .key file, the *onInvokeEvent* function will initialize *Canvas*. |
| 6 | System | *Document* receives a *KEY\_LOADED* event from *ResourceManager* with an Array of data accessed by property: .loadedData.  *Document* passes this data to *Canvas*, which in turn passes it to *LessonContent*, which generates the available games options and displays to the user. |
| 7 | User | User selects a game, which triggers a *GAME\_SELECTION* event to the *Canvas*. *Canvas* will create a new instance of the game and include the storedKeyData as an argument. |

\*Canvas and Lesson Content do not load this file, but may use the file name as part of the UI display.



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **keyData:Array** | | | | | | |
| [0] = lesson code | [1] = available games | [2] = number of images | [3] = images:Array | [4] = game code | [5]images in game | [6] game data: Array |
| “BF1U1L5” | 1 | 6 | [Bitmap][Bitmap][Bitmap]  [Bitmap][Bitmap][Bitmap[ | 0 | 3 | … |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *.key* **:byteArray** | | | | | | |
| lesson code  :UTFBytes | available games  :Float | number of images  :Float | image *n* | game code  :Float | images in game  :Float | game data |
| “BF1U1L5” | 1 | 6 | size of byte array:Float | 0 | 3 | Image ref:Float |
|  |  |  | width of image:Float |  |  | Image mask ref:Float |
| height of image:Float | Initial position X:Float |
| image byte array:ByteArray | Initial position Y:Float |
|  | Target position X:Float |
| Target position Y:Float |
| Initial scale:Float |

Designing a game.

1. Create a game class that will respond to different images or sounds that can be stored in the .key file.
2. Decide what data is needed, including images, coordinates, scaling or other specific game data.
3. Decide how this data will be stored in the key file game data array.
4. Consider how this will be stored in binary format.