EPUBForge - A Simplified Game Engine Based on Features for the Production of Ebook Games

Victor Travassos Sarinho[1]

[1] Laboratório de Entretenimento Digital Aplicado (LEnDA)
Universidade Estadual de Feira de Santana (UEFS)
Feira de Santana, BA – Brasil

vsarinho@uefs.br

Abstract. *This paper presents EPUBForge, a simplified game engine proposal for the production of ebook games (g-books). It is based on provided game features able to represent the gaming factor in a simplified perspective, defining as a result game entities, game assertions, and game scenes with their respective components and events. A game builder framework is also provided to interpret the configured game features and generate valid g-book files according to security limitations defined by ebook readers. As a result, this work provides a reusable game development approach capable of creating g-books that can be played by cross-platform ebook readers and distributed to interested players in EPUB format.*
Keywords *Game engine, EPUB format, Ebook games, G-books.*

1. Introduction

An *electronic book*, also known as an e-book or ebook, is a book publication made available in a digital form, consisting of text and images that can be read on the flat-panel display of computers or other electronic devices [Gardiner e Musto 2010]. Ebooks can be easily distributed on the internet through web pages, social networks, or instant messaging groups. They can also be quickly read on dedicated e-reader devices, as well as on any computational device that presents a controllable viewing screen, including PCs, laptops, tablets, and smartphones.

Game engines represent a common reusable strategy for digital game development, which can be provided by available *SDKs*, *frameworks* and *game libraries* [Gregory 2014], reducing development time and cost [Folmer 2007] to provide them. They can be defined as an “extensible software that can be used as the foundation for many different games without major modification” [Gregory 2014], and represents “the collection of modules of simulation code that do not directly specify the game’s behavior (game logic) or the game’s environment (level data)” [Lewis e Jacobson 2002].

Aiming to integrate Ebook and Game Engine technologies to offer a reusable strategy based on features [Kang 2009] for developing digital games in the EPUB format, this paper introduces EPUBForge, a simplified game engine designed to produce ebook games (*g-books*). EPUBForge utilizes game features that can be interpreted by a dedicated game engine embedded within an EPUB file via a game builder framework, thereby providing valid g-book files according to the security limitations imposed by ebook readers. As a result, by combining configurable game features with the game builder framework, this work presents a reusable game development approach capable of creating g-books compatible with cross-platform ebook readers, which can be distributed in EPUB format to interested players.

2. Related Work

Regarding the production of games for different types of gaming platforms, [Okuda e Emi 2013] demonstrated how to create a game using the EPUB3 format[1] with interactive features based on HTML5-related technologies. From a game engine perspective, after analyzing the features available in free and open tools for creating ebooks, [Figueiredo e Bidarra 2015] presented an evaluation of the feasibility of creating gamified ebooks that effectively facilitate teaching and learning, making as a result a Unity-based tool to provide a novel interactive electronic book called “g-book”. Moreover, according to [Sarinho 2021], GEnEBook is a game engine proposal that employs a game data model and a HTML5 game panel to dynamically represent gamebook adventures, together with a game builder structure that ensures the production of valid ebook files in the EPUB format.

Making a comparison with the proposed work, these papers demonstrate the possibility of developing ebook games in a reusable way. Nevertheless, they presented development problems and feasibility limitations with the EPUB format, HTML5 rendering problems for cross-platform ebook readers, and game design restrictions for other types of game categories. In this sense, the proposed work presents a reusable game development approach able to provide feasible g-books for the EPUB format, without game design restrictions and HTML5 rendering problems for compatible ebook readers.

3. Method

The game engine construction was performed in three main steps. The first was the creation of a feature-based game model able to represent desired games by the definition of game scenes, game components, and game assertives performed by the occurrence of game events. The second step was the adaptation of the Allegro.js[2] game library for EPUB readers, together with a JavaScript+HTML5 game engine code able to execute the modeled JSON in line with the security restrictions imposed by ebook readers. For the third step, which focuses on the production of EPUB files, the GEnEBook game builder approach [Sarinho 2021] was applied, but in this case using the Allegro.js adaptation. This strategy combines existing structures capable of interpreting game configuration files with configured game files, resulting in the generation of valid EPUB files for ebook readers.

Regarding the *feature-based game model*, some features were defined in a proposed JSON game data able to represent desired games, such as *entity*, *assert*, *scene*, *components* and *events* (Figure 1). They are based on common elements and concepts found in available game engines, such as Godot[3] and Unity[4], and represent systematic reusable artifacts able to model the explicit manipulation of system variability [Apel et al. 2016] in a game implementation perspective.

[1]`http://idpf.org/epub/30/`
[2]`http://allegrojs.net/api/`
[3]`https://godotengine.org/`
[4]`https://unity.com/`

In this sense, an *Entity* represents the current game state of designed game objects that can be used by components rendered in active scenes of the game. An Entity can indicate if it is *Active*, *Hidden*, and if it has any graphical information that can be used to render an object in the scene (e.g., *position*, *frames*, *label*, *align*, *border*, *icon*). Entity can also optionally indicate if it has predefined behaviors to be applied when rendered, such as *isDraggable*, *isCollider*, *isPaddable*, *isCamerable* and *isZoneable*.

Asserts are rules that can be executed based on associated events within a game scene, or if they are directly called in a game script code according to the programmed configuration of a game. Each *Assert* has an *Eval* to indicate whether the commands specified in *Exec* or *Else* properties will be executed. If the *Eval* result is true, then *Exec* will be executed; otherwise, the commands in *Else* will be processed.

Each game *Scene* has *Dimensions* and a title that can be used for rendering the game, and it also indicates whether it is the *Active* scene to be rendered during the gameplay. Each *Scene* has a list of *Components* that render the current values of the entities, as well as *Events* that can execute the defined assertives to update the values of the entities.

Components essentially indicate their *Type* (e.g., *Sprite*, *Label*, *Button*, *Timer*, *Zone*, *Dialog*) and the entity that will have the information to be rendered (the *EntityRef*). Each *Event* specifies its name (e.g., *mouseClicked*, *timeout*, *hasCollided*, *frameProcessed*, *keyPressed*) and the *Assert* that will be executed due to the event occurrence (the *AssertToPerform*). If the name of an *Entity* is indicated in the *Event*, its reference will be provided to the *Assert* for desired entity updates according to the specified commands.

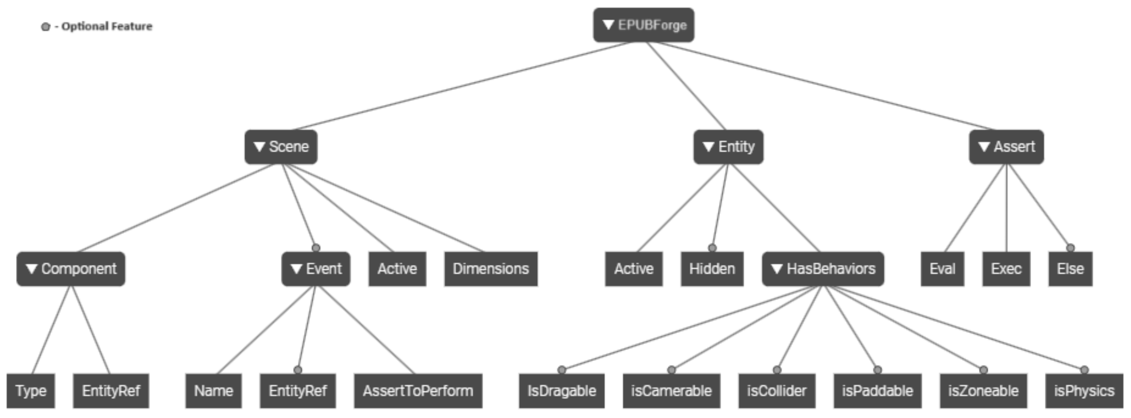
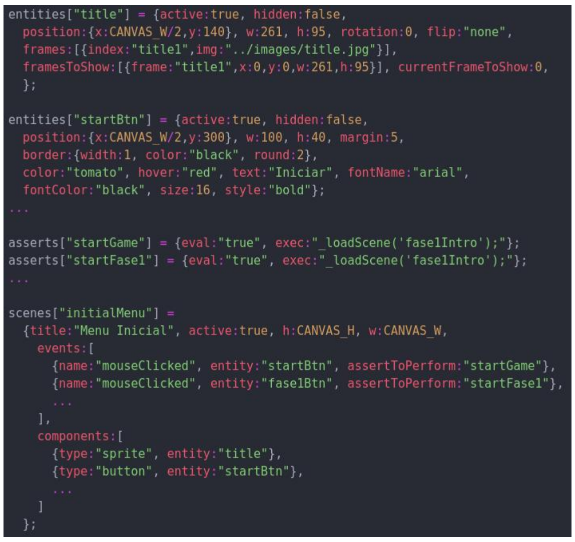


Figure 1. Feature diagram with mandatory and optional features of the proposed JSON game data.

For illustrative purposes, Figure 2 presents a configuration example of: the ***title*** and ***startBtn*** entities; the ***startGame*** and ***startFase1*** assertions; and the ***initialMenu*** scene, which has ***mouseClicked*** events for the ***startBtn*** and ***fase1Btn*** entities that are represented as ***button*** components, together with the ***title*** entity represented as a ***sprite*** component.

Regarding the *game engine* code (the second step), it follows the execution flow of Allegro.js, which performs the *initialize()* function that initializes the game's multimedia resources and sets the initial scene to be loaded, with or without a 2D camera for a specific entity in the scene. Then, the *draw()* method is looped, which performs the rendering of the components in the current scene, together with the *update()* method that monitors the events of the scene's components and the scene itself. For each occurrence of a monitored event, an assertion is executed, which alters the state of the entities rendered by components, thus defining the mechanics and dynamics of the designed game.



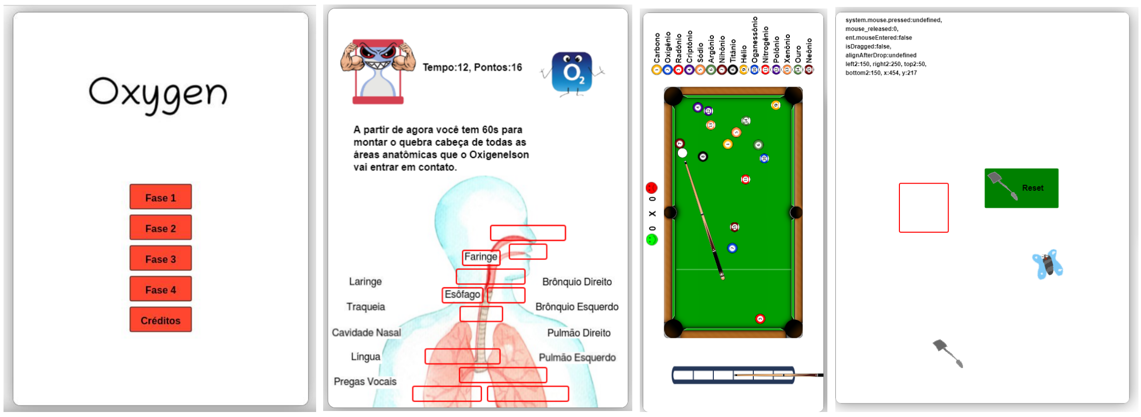
```
entities["title"] = {active:true, hidden:false,
  position:{x:CANVAS_W/2,y:140}, w:261, h:95, rotation:0, flip:"none",
  frames:[{index:"title1",img:"../images/title.jpg"}],
  framesToShow:[{frame:"title1",x:0,y:0,w:261,h:95}], currentFrameToShow:0,
  };

entities["startBtn"] = {active:true, hidden:false,
  position:{x:CANVAS_W/2,y:300}, w:100, h:40, margin:5,
  border:{width:1, color:"black", round:2},
  color:"tomato", hover:"red", text:"Iniciar", fontName:"arial",
  fontColor:"black", size:16, style:"bold"};
...

asserts["startGame"] = {eval:"true", exec:"_loadScene('fase1Intro');"};
asserts["startFase1"] = {eval:"true", exec:"_loadScene('fase1Intro');"};
...

scenes["initialMenu"] =
  {title:"Menu Inicial", active:true, h:CANVAS_H, w:CANVAS_W,
    events:[
      {name:"mouseClicked", entity:"startBtn", assertToPerform:"startGame"},
      {name:"mouseClicked", entity:"fase1Btn", assertToPerform:"startFase1"},
      ...
    ],
    components:[
      {type:"sprite", entity:"title"},
      {type:"button", entity:"startBtn"},
      ...
    ]
  };
```

Figure 2. Partial example of entities, assertions, scene, components and events configuration for the proposed game engine.

4. Obtained Results

Two g-books were developed using the proposed game engine: *Oxygen* and *Snooker of Knowledge* (Figure 3). Oxygen [Bonfanti et al. 2022] is a game that presents mini-puzzles and quizzes focused on teaching and learning the study of human anatomophysiology of the respiratory system in a playful and innovative way. Snooker of Knowledge is a game that applies snooker game mechanics and dynamics to achieve educational challenges, which are presented to the player according to the pocketed ball during the gameplay.

Regarding the Oxygen game, it is the first complete game developed by EPUBForge, which has been evaluated with success by students of a technical nursing course. It is a game that explores features such as: point-and-click, drag and drop of graphical elements, and buttons for quizzes. In this sense, by the collision between objects in a scene, together with the application of Timer events, scene loading, Head-Up Display (HUD) generation, and the use of sound and image resources, Oxygen demonstrates the game engine's ability to develop many types of games with an educational perspective. On the other hand, Snooker of Knowledge, which was developed by two computer science students with no prior knowledge of the proposed game engine, performs the simulation of 2D physics in multiple snooker balls. As a result, it confirms that g-books can be developed by third parties using the proposed game engine, even with the demand for a certain level of performance in its processing. Other features of the game engine are also demonstrated in a tutorial g-book (Figure 3), which presents code structures for the application of 2D cameras on Entities, the presentation of dialog windows with buttons and prompts for data input, monitoring of directional controls for character movement, and the generation of animated sprites with changes in position, rotation and scale.



A partir de agora você tem 60s para montar o quebra cabeça de todas as áreas anatômicas que o Oxigenelson vai entrar em contato.

```
system.mouse.pressed:undefined,
mouse_released:0,
ent.mouseEntered:false
isDragged:false,
alignAfterDrop:undefined
left2:150, right2:250, top2:50,
bottom2:150, x:454, y:217
```

Figure 3. Developed g-books for validation purposes.

Regarding the execution of the developed g-books, success was achieved with the Calibre (v5.41.1) reader for Windows, Thorium (v1.7.4) for Linux, Lithium (v0.24.1) for Android, and Reasily (v2023.05k) for Android. In these e-readers, the g-books worked as expected in terms of performance, graphic rendering, image positioning, user input processing, and audio output generation. However, some problems with the initial upload of game assets (e.g., sound and image) were presented by the Allegro.js when web-based EPUB readers (e.g., EPUBjs[5], EPubReader[6]) were used. It is a consequence of the varied resource access pathways used by these readers, highlighting the need to define a more suitable approach for allocating the game resources that they will load.

5. Conclusions and Future Work

This paper presented EPUBForge, a game engine proposal able to provide ebook games (*g-books*). Among the benefits: it has the ability to model games for different categories in EPUB format, instead of working with a specific game style (e.g., text adventures, quizzes); it offers reusable components represented as features (e.g., Camera2D, Draggable, Zoneable) that are not provided by game libraries such as Allegro.js; and it provides the correct rendering of the modeled content for games, without graphical variations presented by distinct HTML5 rendering in different EPUB readers.

However, despite the successfull e-reading for mobile and desktop platforms, some problems were found with web readers that need to be solved in the future. Aspects such as: reengineering the developed engine architecture; developing new game components (e.g., sliders, menus, stream loader); and implementing new entity behaviors (e.g., *isPhysics*, *isIntelligent*, *isMultiplayer*) also need to be addressed in the future. Finally, regarding the use of a low-code programming strategy with the proposed game engine, future adaptations for block-based programming environments (e.g., Arcade MakeCode[7], Blockly[8]), together with the development of low-code configuration tools for specific game categories (e.g., Kahoot[9], WordWall[10]), will be performed in the future.

[5] `http://futurepress.github.io/epub.js/examples/input.html`
[6] `https://www.epubread.com/`
[7] `https://arcade.makecode.com/`
[8] `https://developers.google.com/blockly`
[9] `https://kahoot.com/`
[10] `https://wordwall.net/`

References

Apel, S., Batory, D., Kästner, C., e Saake, G. (2016). *Feature-oriented software product lines*. Springer.

Bonfanti, F., Gutierrez, L., Maciel, V., e Sarinho, V. (2022). Oxygen-um gamebook para o estudo da anatomofisiologia humana do sistema respiratório. In *Anais Estendidos do XXI Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 1336–1340. SBC.

Figueiredo, M. e Bidarra, J. (2015). The development of a gamebook for education. *Procedia Computer Science*, 67:322–331.

Folmer, E. (2007). Component based game development: A solution to escalating costs and expanding deadlines? In *Proceedings of the 10th International Conference on Component-based Software Engineering*, CBSE'07, pages 66–73, Berlin, Heidelberg. Springer-Verlag.

Gardiner, E. e Musto, R. G. (2010). The electronic book. *The Oxford companion to the book*, page 164.

Gregory, J. (2014). *Game engine architecture*. AK Peters/CRC Press.

Kang, K. C. (2009). Foda: Twenty years of perspective on feature models. *the keynote of SPLC*.

Lewis, M. e Jacobson, J. (2002). Game engines in scientific research. In *Communications of the ACM*, volume 45(1), page 21.

Okuda, S. e Emi, K. (2013). Make once, play anywhere!: Epub 3 interactive function enables us to make and play game software anywhere! In *2013 IEEE 2nd Global Conference on Consumer Electronics (GCCE)*, pages 381–384. IEEE.

Sarinho, V. T. (2021). Genebook: A game engine to provide electronic gamebooks for adventure games. In *2021 20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 59–68. IEEE.