CONCEPT AND IMPLEMENTATION OF A WEB BASED TOOL FOR THE DEVELOPMENT OF ONLINE GAMES FOR EARLY CHILDHOOD EDUCATORS

B. Penz, J. Mäkiö
University of Applied Sciences Emden / Leer (GERMANY)

Abstract

Today, the transfer of digital skills is becoming increasingly important. In the area of early childhood education special attention must be given to a critical-reflective approach, so that children learn the limits of digital media and develop the ability to control their media consumption. Educational applications may help children acquire these skills. Early childhood educators often lack the technical skills to independently develop digital applications. Some modern tools provide programming beginners with graphical programming an introduction to software development. These tools hardly go beyond the pure learning approach. Thus, an application development tool that is suitable for early childhood education is needed. This paper deals with the prototypical development of a web-based game development tool for early childhood educators. A human-centered design process was applied to achieve a good user experience for the target group. A group discussion was organized early with students of childhood education to identify their needs and to gather ideas for the application. Based on the requirements and a discussion of alternatives, a new tool was designed and implemented. The tool supports the concept of graphical programming, in which symbols are combined in a simple if-then system to produce dynamic behavior and the naming of concepts avoided technical terminology. In the final focus group test a good user experience for the temporal dimension before use was determined. The basic concepts of the application were well understood. Only four out of 31 participants did not dare to use the application at the end of the test. An online survey was used to evaluate the user experience. Students were able to try the tool at home for this purpose. Here, a mixed user experience and improvement potential of the application was found.

Keywords: early childhood education, online game.

1 INTRODUCTION

The digitalization of society is increasingly changing people’s lives and working lives. 12 percent of jobs in Germany have a high probability of being automated in the next few years [1]. In the United States, 47% of employees are likely to be replaced by machines in the next 10 to 20 years with a probability of more than 70%. Digital skills of employees and the transfer of digital skills is becoming increasingly important. Not only adults, but also children and adolescents use digital media more and more often [2]. According to studies in the US, the number of primary school children playing video games has increased [3] [4]. On average, young people from 8 to 18 play about one and a half hour a day [5]. Another study conducted with schoolchildren in Singapore has found that more than 80% of the children played video games and each child played an average of 20 - 22 hours per week [2].

Playing video games has different long-term effects. It has been shown that action games can be beneficial to several adult spatial-visual abilities. For example, the visual-spatial attention that enables the efficient tracking of individual objects in a group of multiple distracting objects improves [6]. Another example is the enhancement of the spatial-visual resolution that can be used to identify small objects in a collection of multiple objects [7]. In [3] similar effects in children aged 6 to 12 years are shown.

Video players usually develop better visual short-term memory [8] and they may become more flexible in changing between various tasks required in the game. Several studies confirm a connection between playing video games and improvement of various cognitive abilities of the players [9]. Children who play video games have significantly improved attention skills compared to children who do not play [10]. In addition, the interactive multimedia environments of video games with their images, animations and sounds may have a stimulating effect on the learning interest of children [11].

According to [12], by using a multimedia serious game, kindergarten children could better remember what they had learned. An attempt was made to make the user interface of the software child-friendly,
with many colors, funny motives and music [12]. Also, language skills of children can be promoted with the help of video games. According to [13], alphabetic vocalizations of preschoolers could be improved by means of a specially developed video game system.

However, the playing of video games has negative effects, too. For example, it has been found that 8% of children who play video games develop pathological usage patterns. This often interferes with the performance at school and various cognitive abilities [4]. Playing violent video games can also increase aggression in children. A meta-analysis of several research papers has shown that playing video games can increase aggression in children while also leading to anti-social behavior and decreased empathy [14]. A study shows that some players of action videogames experienced grey matter loss in their hippocampus after playing action video games for 90 hours. Players who are ‘spatial learners’ increased their grey matter after playing the same amount of time [15].

Bearing in mind the possible negative effects of video games in the area of early childhood education special attention needs to be given to a critical-reflective approach, so that children know the limits of digital media and develop the ability to control their media consumption. Educational applications could help children acquire these skills. Early childhood educators should be able to develop digital applications that playfully assist children in acquiring and promoting digital literacy. However, programming is a complex and time-consuming task, requiring skills and concepts that are usually new to the learner [16] [17], and early childhood educators are often technically not well-versed or interested in learning programming skills. In addition, programming courses often have high dropout rates because they are perceived as difficult and boring [18]. Also, many teachers tend to provide theoretical knowledge than teaching practical skills [19], although programming has more to do with designing and execution than with knowing theoretical facts [20].

To tackle these problems, there is a need for a system that allows application development without requiring classical programming skills. One promising approach is the Graphical Programming involving a variety of features: animations, graphic representation of data and tools in graphical form [21]. Graphical programming languages allow the programmer to create applications with graphical expressions and the spatial arrangement of text and graphic elements [22]. The graphical programming seems to be the most advisable approach when developing programming tools for non-programmers.

A variety of graphical programming languages and programming environments have been emerged for various purposes, like visualization of data, control of industrial and business processes, game development, and sophisticated general-purpose languages that are not limited to any specific domain [23] [24]. One of the first attempts is the programming language HI-VISUAL, that supports visual programming by arranging icons that represent functions and data [25]. HeNCE is a graphical programming environment for learning parallel programming [26]. Other environments based on graphical programming followed. For instance, the Lego Mindstorm NXT robots feature an icon-based programming of physical Lego robots [18]. RoboMind is a graphical programming environment for learning programming by manipulating the behavior of a simulated robot [23]. The Lifelong Kindergarten Group at MIT develops the visual programming environment Scratch. The software is freely usable and is specifically designed for teaching programming to children from the age of 8 to 16 [16].

This paper describes the conception and implementation of a web-based application that allows early childhood educators to develop games without having classical programming language skills by relying on graphical programming techniques. The application is geared to the needs of early childhood educators in order to increase the readiness to use the target group.

2 METHODOLOGY

The method Human-Centered Design (HCD) was applied to design the application, as the focus in the HCD is put on users' needs. IN HCD both pragmatic and hedonic (like joy and attractiveness) qualities of an application are considered simultaneously. This allows to achieve a good user experience and to improve the product acceptance [27]. This is required as the target group is lacking on technical skills.

The HCD comprises the requirements analysis and requirements definition based on the target group’s needs. Thus, the HCD was applied with the goal to achieve a good user experience, and because of that, a high user acceptance. For this purpose, 1) a group discussion was organized with students of childhood education to identify their needs and desires to gather ideas for the functionality of the application. Based on the result from the group discussion, a concept for a 2) a new web based
tool for early childhood educators to create online games was designed and implemented. Finally, 3) the user experience of the target group was analyzed in two steps: a) demonstration of the tool by means of practical exercises and subsequent and b) online survey to gather further ideas for the tool improvement, respectively the prototype’s user experience.

2.1 Game Engines and tools for game development

To determine basic functionalities needed in the tool to be developed, four game engines (Unreal Engine 4, Unity Engine, Game Maker Studio 2, and Scratch) were analyzed regarding their characteristics, strengths and weaknesses. Subsequently, a distinction has been made to the new tool developed.

The analysis shows, that the Unreal Engine and the Unity Engine both have similar features. They are aimed at more tech-savvy users and professional users. The strengths of Scratch are in the education. Thus, it is not suitable for application development because important functions such as switching scenes or scrolling over the game world are missing. Game Maker Studio 2 is a compromise. It is aimed for beginners and it requires programming in a classic manner. In addition, the user interface is complex and thus rather inhibiting for technically inexperienced users. None of the examined applications are applicable as a game development tool for early childhood educators. However, some characteristics of the examined tools are applicable. For example, the basic structure of the user interfaces of Unreal Engine 4 and Unity Engine is suitable. The structure includes navigation bar with game objects, menu bar, preview area, scene overview, detail view and content area. The basic structure can be adopted for the new tool. The basic concept of graphical programming like Scratch may apply to childhood educators but it needs to be tailored to their needs.

2.2 Group Discussions

To establish a baseline for gathering requirements for the tool, a group discussion was organized in the Hochschule Emden / Leer with 11 female and two male students of early childhood pedagogy and the authors as moderator. The goal of the discussion was to generate ideas for the tool. To create a basis knowledge level for discussion, the moderator explained at the beginning the concept of a game engine using as an example the game engine “Unity”. At the end, the results from the group discussion were put together with the results from analysis of existing game engines (Unreal Engine 4, Unity Engine, Game Maker Studio 2, Scratch) that were selected based on their market relevance or their focus on beginners. The collected ideas are divided into three categories:

- **Game Ideas**: The suggestions for games to be developed with the tool give hints about the functionalities that should be supported by the new tool, and which templates should accompany the tool. Suggested game ideas were in particular: home designer, drawing program, picture book editor, quiz game, multimedia diary, exploration game, dexterity game.

- **Guiding principles for games**: As the generated games should be educative for small children, the group generated the following the guiding principles: Gender-neutral design, no rhythm games, not too much noise and music, game worlds like real world environments, game worlds based on fantasy environments, no general game objectives for exploration games, prevention of addictive mechanisms like high scores and gamification, spoken text to help children operate the program, games should convey media literacy.

- **Features of the tool**: To be beneficial for the user the tool should have a wizard feature that assists the user in the operation of the tool and guides it through the program. Additionally, it should have interactive tutorial or video tutorial and a good documentation. The graphical user interface should apply tool tips for common functionalities that should be displayed as soon as the mouse pointer is moved over them. The names of controls and categories should be based on the mental model of the target group childhood educators. Technical is to be avoided. The user interface should be easy to use. The programming of the game logic should be intuitive and non-technical. Simple if-then designs should be graphically constructible without having to write source code. The tool should provide game templates as skeletons providing a simplified easy-to-understand introduction to game development and to the functionality of the tool. The tool should support 2D graphics and 3D graphics. 3D graphics match the viewing habits of many children growing up with the consumption of modern animation graphics.

The requirements of the goal group need to be defined prior to the actual development. The requirements need be translated into the design and implementation of the tool. A distinction between
functional and non-functional requirements is to be made. The requirements have been derived from
the previously analyzed game engines and the group discussion with students of childhood education.

✦ Functional Requirements

The basic structure of the user interface is based on the game engines discussed in Section 2.1. The
participants in the group discussion agreed to the layout in the navigation bar with game elements,
menu bar, preview area, scene overview, detail view and content area. The group discussion has
shown that 2D graphics are in most cases enough and that the willingness to acquire required skills is
low. The development of 3D games requires more effort and know-how. Thus, the prototype supports
only 2D graphics. The most important functional requirements are as followed: It must be possible to
add and to delete game elements. Game elements should have a changeable name, properties for
size, position, colour, speed. The properties of a game item must be displayed and defined in a details
view. Sprites must exist as a special kind of game elements. They allow animations. The detection of
collisions must be supported. Certain game elements need to be able to respond to events with
actions. The definition is made by graphical programming: events and actions are added, deleted,
positioned and manipulated. Applications need to be loaded and saved. Debugging messages must
be output in a console. There must be a scene overview to inspect the entire game. There must be a
preview view showing the game in real time from the player’s perspective. Two game templates are
needed: 1) quiz game template and 2) action game template. A game must have various game
worlds, between which can be changed.

✦ Non-Functional Requirements

The non-functional requirements arise as direct results from the group discussion. They are as follows:
The templates should be gender-neutral. The templates must not contain blood or violence. The
templates should not have too much noise and do without too much music. Game content must be
age appropriate. Play worlds can be borrowed from reality or fantasy worlds with, for example,
dragons. There should be no gamification elements such as high scores or Achievements. The names
of controls and categories should be consistent with the mental model of early childhood educators
and should not be IT-specific. The operation of the editor should not require any programming
knowledge. The development effort for the user should be as low as possible.

3 RESULTS

This section describes the development and evaluation of the new tool. The tool consists of two main
components: The runtime engine works in the background to execute the built games and perform
real-time features for the development process. The second component is the editor, which serves as
the interface between the runtime-engine and the user. The editor supports child educators in the
development of games by utilizing easy to understand concepts like drag-and-drop, graphical
programming and names (for concepts and elements) that are supposedly according to the mental
model of child educators.

3.1 Conception of the Runtime Engine

Global variables are used to storage data for changing game sections and for the game control. They
are valid beyond the spaces and can be globally manipulated and read. The data type of the global
variable is “number”. For binary operations, the variable can simply be set to 1 or 0. In addition, there
are occasionally time-dependent actions that are controlled by a timer. A timer has a start time and a
current time, and it may have three states: “Running”, “Paused” and “Finished”.

Graphics and animations allow to create visually appealing sceneries. They are defined by so-called
“stamps”. A stamp can be assigned several 2D graphics that can be played as an animation. It has a
definable color value. The coloring is used when no graphic is assigned to the stamp. An animation is
played if the stamp contains more than one graphic. The stamp has a definable frame time that
determines the speed of the animation. Stamps correspond to sprites known from 2D graphics. The
term "stamps" was chosen because they can be stamped on the background of a scene.

Robots are program elements that enable user interactions and automated processes. They are used
to create representatives in the scene. Robot can be, for example, the character of the player,
computer-controlled characters, objects or obstacles. If the graphic representation of the robot is a
picture or an animation, then a corresponding stamp must be added to the robot. The robot then
assumes the visual appearance of the stamp. Robot stickers can broadcast messages to other robotic
stickers. For the sake of simplicity, the definition of robot behavior is based on simple if-then constructions. Robot events concern messages between objects, mouse events, collision of objects, changes of global variables, and timer expiry. The events are used to trigger actions.

The game world is divided into sections (called “rooms”). The game world consists of at least one room. Rooms can be changed at any time by triggering the appropriate action. Stamps and robot can be added to a room to define the level or menu layout. Rooms can be provided with a background graphic or background color to make the game environment more appealing. The built game is visually presented to the user. For this, the renderer generates a rasterized 2D image of the current scene. The nature of the image depends on the background image, the properties of the stickers (visibility, position, rotation) and the position of the virtual camera. The virtual camera projects a section of the game scene orthographically onto the image plane and is defined by height, width and position in 2D coordinates. The rendered image is drawn on an HTML5 canvas.

The game elements need initial start values and the current game action is calculated by the Game Processor. After starting the Game Processor, the game elements are loaded and initialized. The tool has two operation modes of operation: Developer mode for drag-and-drop of elements, and game mode for running the game in preview mode or as stand-alone.

3.2 Conception of the Graphical Editor

The graphical editor interface is the interface between the runtime engine and the game developer. Its structure is oriented towards media management, navigable bar and a detail view on the interface of established game engines. A unique feature is the area for graphical programming.

The navigation area allows to navigate, create and delete game elements. Game elements are listed by name and they can be removed form from the application and the list. Their details can be displayed and edited. The areas can be configured using a configuration window in which the name of the new element and the type of the element can be determined. The detail view shows the manipulable properties of the selected room, robot or stamp. The view shows the properties of the currently selected game element.

The view of the current room shows a graphic section of the room. With buttons for the directions top, bottom, left and right can be navigated through the room with the entered step size in the appropriate direction. The position of the rooms can be entered directly. Robots and stamps can be selected directly with the mouse and positioned in one place.

The interactive preview shows the game world from the perspective of the player. This allows the user to try out his game instantly via the editor, which speeds development time. Once the preview has been started, the game can be played. Feedback about actions are shown at the console allowing the user to understand his/her work steps. A message consists of the date and time and the content.

Prior to their use, pictures and audio files need to be added to the tool via the media manager panel. The programming of the robots is done graphically to meet the needs of the target group. The graphical programming area is divided in the two sections: events and actions. The individual events and actions are assigned to categories for clarity. The individual categories can be selected by pressing the corresponding tab. In the left section, events are dragged and dropped from the top bar into the event area to place them. There are the following categories for events: Input - for interactions with the mouse and Observation - for perception of the environment (for example, collisions, receipt of messages). Events can be connected by activating a checkbox. All events must then occur for the actions to be triggered. If the checkbox is not selected, at least one event is enough. In the right section the actions are placed. The menu bar above the action area has three tabs for the categories: Movement - for instant positional changes and linear motion; Rotation - for rotational movements and angular velocities; Visual - for aspects of presentation (for example color change, visibility); Audio - for playing audio files; and Logic - for logical operations (e.g. manipulating variables, sending messages). Directly below are the actions of the currently selected category. The actions are dragged and dropped into the action area.

3.3 Conception of the Templates

Two templates were designed to serve the users of the tool as a starting point for their own developments. In addition, they should help users to learn how the tool works. Both templates together represent all concepts of the tool. The templates can be used as an example, when developing own, more complex games.
The first template is a quiz game that uses simple symbols and words so that children should be able to handle it. In the template the player should determine whether a shown object is a fruit or a vegetable. The second template is an action game in which the player should control the character. In the template gender of the figure is not recognizable. The goal of the game is to reach the target area at the top of the screen without being touched by any of the birds. The character is controlled by the arrow keys that are displayed on the screen. Game figure and birds are animated by stamps.

3.4 Implementation

For elementary arithmetic operations on vectors and matrices, the library gl-matrix was used [28]. The rendering of a scene is implemented using the Three.js library that creates a render object that is connected to an HTML5 canvas element [29]. The React JavaScript framework was used for creating GUI. A React GUI is divided into reusable components that are hierarchically ordered. The components can be identified as HTML tags in other components. Each component has a render method that returns HTML code to the calling component. The components can manage their own state and allow unidirectional data flow through the component hierarchy [30]. Reflux is a Flux architecture-based JavaScript library [31] that extends React's unidirectional data flow. Data from multiple components is shared through a store. [32]. Bootstrap is a frontend framework for fast and easy web development. Bootstrap contains several CSS and HTML based design templates for HTML elements. Bootstrap supports Responsive Design which allows the automatic adaptation of the web page to different screen formats [33].

Figure 1. The Graphical Editor of the tool.

The runtime engine consists of a few components. One component that is responsible for the implementation of the game world is a container for specific game objects (Robots, Stamps). The global variables and time based events are encapsulated in separate components. For the information exchange between the game objects, a messaging system component was developed. It receives messages and is responsible for their forwarding to the receivers. To control the animations in the game world, a specific component was developed. An event-system implements a few system events and user actions enabling system components to react on certain events, like mouse movements and timer expiration. Mouse-interactions are handled by functions using raycasting capabilities of the graphics library Three.js to determine whether an object is intersected by the mouse cursor. Further, the tool provides an array of camera objects that shows during the development for the developer the rooms of a game world as it will be shown during the execution of the game. Another array of camera objects allows navigation through the rooms during development. Built game worlds are stored in JSON format.

The user interface is divided into various React components. The visual representation is implemented using the Bootstrap library. The administration of the data takes place via a reflux architecture.
Fig. 1 depicts the graphical editor with the loaded action-game template. The navigation bar is on the left. The menu bar is on top. The right side of the figure shows the details panel. In the middle of the editor, there are the cameras for room overview and game preview. The room overview section also has a button-interface for room navigation. The bottom panel has tabs for the console, media management and the interface for graphical programming “behaviour”. Icons for events and actions can be moved to the according area by drag-and-drop. Like all other elements, events and actions can be manipulated by the inspector after clicking on the respective icon.

3.5 Evaluation

To evaluate the user experience of the new tool, a test with a focus group was organized. The participants were students of childhood education. For the testing and for the filling of the prepared questionnaire, few days were given.

3.5.1 Focus group testing

In the focus group testing, the user experience of the developed tool was evaluated. 31 students of early childhood pedagogy participated (3 male students, 28 female students). Four students have completed training as educators or social assistants. Five students have no kindergarten experience. From these students one has experience in elementary school. Two have experiences as au pair and one has experiences with own children. One person has no experience with children at all. One of the authors moderated the session.

During the demonstration participants were asked about their impressions. If a participant made a statement, the other participants were asked if they agreed with it. The approval was mostly signalled by nods and utterance of the word "yes" of the group participants. If the result was uncertain, the participants were invited to contradict the statement. To determine to what extent the tool is self-explanatory, the participants were invited to first derive the functionality of the different areas by themselves. Subsequently the moderator explained the system functionalities. At the end of the event quantitative data by questioning the participants was collected.

The results point to the following: The overall goal to create a tool that is understandable and usable by early childhood educators has been achieved. The type of user interaction was perceived by the participants as positive. The terminology and the labelling need improvement. The comments of the participants indicate a good hedonic quality of the application. The user interface was perceived as aesthetic. Overall, the colour scheme was positively received. The interactions were also perceived as positive. Some questions were asked to the participants to collect quantitative data. The participants were asked for a hand sign if they agreed to the question. After the demonstration, over 22% of the participants can imagine using the tool to generate a game and to play with the game with children. 13% of the participants felt unsure about using the tool to develop a game and 87% could imagine creating games with the tool. Around 39 % say to have a concrete idea for a game they want to develop with the tool.

Considering that the effort for game development is relatively high and the target group is generally considered to be technically not very interested, the answers point to a positive anticipatory effect of the application on the participants. This fits in with the pragmatic and hedonic quality of the application noted above. It is noteworthy that only 13% of the participants do not trust themselves to make use of the new tool.

3.5.2 Online survey

To supplement the test with the focus group, a survey took place between March 7, 2019 and March 12, 2019. 15 students of early childhood pedagogy participated in the survey. In the focus group, only the anticipatory effect of the application on the target group was tested. To cover the period after use, participants also had the opportunity to test the application at home. For this purpose, the application was uploaded together with the templates on a web presence. Subsequently, the students were able to participate in a questionnaire. The goal of the questionnaire was to determine the pragmatic quality and hedonic quality of the application.

Responses were reduced to key statements to keep the analysis clear. Similar statements have been summarized and weighted. Finally, the statements were divided into the categories "positive utterance" and "negative utterance".
The answers show that many aspects of the engine were often well understood by participants. In particular, the operation of the stamp and navigation arrows was understandable. The user interface was found by some to be clear. Others found the user interface initially confusing. Some people had the impression of having understood the application during the focus group testing, but when they tried it out at home, they realized that they have not understood the application. Here is a discrepancy between anticipation and time after use. However, a large proportion of the interviewees assumed that there should no longer be any problems in operating the application with an introductory help. Overall, however, the positive impression also prevails here. Given that the visual design was functional, expectations are exceeded. Trying out the features made some respondents happy. Other interviewees saw potential that the fun could be set as soon as the first sense of achievement comes.

The application was able to stimulate interest in game development among many respondents. This corresponds to the results of the focus group test. Many respondents said that the time spent building applications was too high. In this respect, simplification of the application should be achieved in future work. Other obstacles were the operation of the program and a lack of technical interest.

4 CONCLUSIONS

Participants were impressed by the concept of the Stamp system, the concept of graphical programming, the choice of graphical symbols and the visual design of the user interface. The time was spent on game development, missing or confusing labels and bugs. Thus, the objective was to evaluate the new tool in terms of their user experience.

As part of future work, the application could be improved. This would make it possible to optimize the performance of the application to increase the refresh rate of the graphic output. Furthermore, an interactive introduction to the operation of the application could be developed to simplify the operation. The survey participants have criticized the lack of such a function. In addition, it should be considered to review some terms of the concepts of the new tool. It would then be possible to replace the old concepts with terms that prove more suitable for the mental model of the target group. In particular, the term “robots” led to some confusion in some participants of the evaluation or was perceived as not nice.

Many participants complained about the time needed to develop games. It should be determined how a meaningful time saving can be achieved. One possibility would be to provide the engine with new events or actions that map out more complex behavioural structures, such as the jumping of a paw. These would then no longer have to be simulated via other events or actions.

It can also be checked whether the new tool is also suitable for other target groups. It is conceivable that schoolchildren learn the basics of algorithmization / programming with the new tool. To reach younger children, the development of a simplified version of the tool would be possible.

ACKNOWLEDGEMENTS

We would like to thank Prof. Lena S. Kaiser and her students from the University of Applied Sciences Emden / Leer for her support and cooperation in the preparation of this work.

REFERENCES


[31] “Reflux Documentation”, URL: https://github.com/reflux/refluxjs/tree/master/docs, Abruf am 05. 02. 2019


[33] „Bootstrap 4 Get Started“, URL: https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp -Abruf am 05. 02. 2019