Supporting CS1 Instructors: Design and Evaluation of a Game Generator

Jecton Tocho Anyango*
Department of Computer Science,
University of Cape Town,
Private Bag X3, Rondebosch 7701
Cape Town, South Africa
anyjec001@myuct.ac.za

Hussein Suleman
Department of Computer Science,
University of Cape Town
Private bag X3, Rondebosch 7701
Cape Town, South Africa
hussein@cs.ac.za

ABSTRACT

Serious games have shown much promise in education, including in the teaching of programming. However, instructors who teach introductory programming often do not have the specialised skills to create serious games. One way to address this problem is to use domain-specific game generators to create customised games as needed. This paper presents the design and empirical evaluation of a prototype game generator tool - the Recursive Game Generator. 30 programming instructors evaluated the tool and found it useful (87%), easy to use and learn (80%); and were satisfied with the tool's effectiveness and efficiency. Their positive experiences suggest that such a higher-order tool has the potential to increase the adoption of serious games in programming education, and broadly meet the needs of a diverse audience of instructors.

CCS CONCEPTS

• **Social and professional topics → Computer science education**; CS1.

KEYWORDS

Introductory programming; CS1; game design; game generator; usability; recursion; novices; instructors

ACM Reference Format:
Jecton Tocho Anyango and Hussein Suleman. 2021. Supporting CS1 Instructors: Design and Evaluation of a Game Generator. In *Proceedings of the 2021 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '21), June 26-July 1, 2021, Virtual Event, Germany.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3430665.3456306

1 INTRODUCTION

Within the Computer Science Education(CSE) community, research has shown that programming novices and teachers continue to face serious difficulties and challenges [14] [17]. We use the recursion topic as a case study in this paper given its difficulty among students and the demonstrated potential of using games to teach the concept [5] [6]. Educators often look to technology to support their teaching. To this end, some scholars have suggested Technology Enhanced Interventions (TEI) such as game authoring platforms [18] [24]. However, most current game authoring tools and technology platforms are commercial and target specialised developers or game programmers [32]. Meanwhile, developing serious games is time consuming, difficult and expensive - hindering teachers who may wish to adopt GBL in their mainstream teaching [31]. Serious games are games designed for use in contexts other than entertainment or to achieve goals other than entertainment such as education, health, cultural heritage e.t.c [12, 19]. They are games designed to educate, train, and inform [11]

The purpose of the current study is to explore the user experiences of programming instructors when using a prototype game generator tool - the Recursive Game Generator (RGG) as a means of creating serious games. The focus is on a prototype to support CS1 educators. It is not about evaluating improvement in student learning outcome or knowledge gained. Consequently, the primary respondents in the study are higher education programming instructors who are the target beneficiaries of the proposed idea of the tool.

To address the study purpose, this paper reports on evaluation of the usability of a prototype game generator tool - the Recursive Game Generator (RGG) with local teachers by answering three research questions:

(1) How useful and easy to use do programming instructors from Kenya and South Africa find the RGG generator tool?
(2) What is the opinion of programming instructors from Kenya and South Africa about the RGG tool's learnability?
(3) How satisfied are programming instructors from Kenya and South Africa with the RGG generator tool?

According to Sauro and Lewis [27], usability refers to the extent to which a system can be used by specified users (programming instructors in this study) to achieve specified goals with effectiveness, efficiency and satisfaction in a given context of use. To these, a leading scholar in usability studies adds memorability, learnability, and few errors [21]. This work is guided by these 2 definitions.

The main contribution of this paper is therefore that it provides empirical evidence of the suitability of a prototype game generator tool for supporting instructors to easily create games to teach CS1.

The rest of the paper is organised as follows: Section 2 presents related work. Section 3 describes the tool's prototype design and development. Experimental design is presented in section 4 while results are in section 5. Discussion is in section 6. Lastly, section 7 presents conclusions and future work.

*Both authors contributed equally to this research.

2 RELATED WORK

Studies on serious games authoring tools are becoming increasingly relevant among different education research communities. For instance, Bouzid et al. [4] suggested a game generator tool capable of authoring several instances of the MemoSign game for deaf learners. On the other hand, a semi automatic generator of tactile video games to help visually impaired children was proposed by Sepchat et al. [28]. Recently, the SHARE-IT project was designed to support parents and teachers of autistic children [24]. Although both the MemoSign game generator and the SHARE-IT project were worthy efforts in the right direction, they were not empirically evaluated. The uAdventure toolkit [25] is an improvement of the e-adventure platform [30] and is built on top of the Unity game engine. Both are instructor oriented tools for authoring video games for e-learning materials like books. Consequently, they may not be adapted for programming games. We extend the work by Perez-Colado et al. [25] by building RGG on top of the Unity game engine but the proposed tool authors programming games.

Within the CSE community too, authoring toolkits have started featuring prominently as a useful area of research as illustrated in the Framework for Gamified Programming Education (FGPE) [23]. However, the project seems to be at the conceptual stage. Khenissi et al. implemented the Learning version of Pacman Game Generator (LPG) and the Instruction Right Place Game Generator (IRPG) for teachers [11]. Although the generated games were aimed at teaching programming, the focus was on the Mapple programming language. The games generated in the current work allow students to practice programming in Python - the most popular language for teaching CS1 [29].

Regarding evaluating serious game authoring tools, some authors have suggested that empirical evaluation should be based on how well the generated games score against the generation criteria [22]. These criteria include the targets set by the tool designers such as: (i) game quality; (ii) game inventiveness; (iii) game video/audio excellence; (iv) game length; (v) constraints on solvability; (vi) game mechanics or (vii) any other design parameters. Another study identified usability evaluation criteria for game authoring tools for teachers as: (i) easy of learning; (ii) error rate; (iii) efficacy; and (iv) efficiency [3].

The work by Machiori et al. presents the Writing Environment for Educational Video games (WEEV) system - an instructor oriented game authoring tool [18]. Two formative evaluations - one with 9 teachers and another with 20 students revealed that the tool was complex to use. Particulary, users singled out understandability and learnability usability problems. The study by Gaeta et al. [7] evaluated the usability of an authoring tool for creating Storytelling Complex Learning Objects (SCLOs) in the ALICE project. A System Usability Score (SUS) of 60.625 was reported, which could be considered relatively lower than the satisfactory threshold of 68 with minimum score 45 and maximum 77.5. In computing education, Khenissi et al. [11] conducted an experiment with 167 first grade students to evaluate the effectiveness of serious games created using the IRPG and LPG generators compared to learning versions of existing games. Results suggested a positive impact on student knowledge gain when using the generated serious games. However, students cannot practice coding with the games generated by the LPG tool.

In summary, despite the educational potential of serious games, not much work has been done in the area of game authoring in computing education. Particularly, it appears limited empirical evaluation has been conducted with educators (primary users). The current study extends existing works by proposing the idea of a programming game authoring tool and evaluates a prototype with CS educators.

3 TOOL DESIGN AND DEVELOPMENT

3.1 Design Goal

The design goal of this project was to develop a usable prototype to support programming instructors with no game programming skills to easily create serious games to teach novices programming in CS1.

3.2 Design theory

The design of the proposed tool is guided by two theories: (i) differentiating interfaces theory; and (ii) the complexity theory [10]. Differentiating interfaces ensures the design of different interfaces/user profiles for different users (novices and advanced users). On the other hand, complexity theory hides the complexities of the underlying technologies from the users (teachers).

3.3 Design Method

Some previous works have explored methods to guide development of serious games [8] [26] . For instance, Ibrahim and Jaar [8] proposed a model comprising game design, pedagogy and learning content modelling. On the other hand, Saavedra [26] suggested a software process to game design comprising 5 steps: (i) requirements gathering - setting learning goals; (ii) design - creating digital resources; (iii) development - creating the game; (iv) testing; and (v) postmortem. This proposal best suits large scale commercial serious games but may not be practical for learning institutions. The design process of the prototype game generator tool followed the User Centered Design (UCD) method. We adopted three key UCD design elements:

(1) User participation
Two types of users (primary and secondary) are suggested by Maguire et al. [16]. In this study, CS1 instructors are the primary users (participants) while students are secondary users.
(2) Contextual inquiry
This involves considering the users'work needs in context. We considered contextual elements such as difficult CS1 topics, time allocated to the topic of recursion by the curriculum, diversity among CS1 learners in the context of Kenya and South Africa, recursion topic teaching scenarios, instructor game programming skills and game duration.
(3) Iterative design
We established early contacts with primary users through a needs assessment study with CS1 lecturers [2]. This was followed by a conceptual qualitative user requirements study.

During development, we continuously focused on user requirements and iteratively tested the tool with 2 experienced volunteer CS1 educators from the department of computer science of the university. The two acted as co-developers by means of cooperation in a non-linear and iterative manner [33].

We adapted a game design methodology from the early works by the Game Development for Computer Science Education working group of Innovation and Technology in Computer Science Education 2016 (ITiCSE 2016) [9] and Saavedra [26] in the design of the prototype of a Web-based game generator tool called RGG[1].

3.4 Game generation

Game generation process occurs in 9 steps as follows:

- Step 1: Select how you want to create a game (using an existing example or from scratch or using a progress file)
- Step 2: Select how the player will view the game
- Step 3: Select the scene you want in the game environment
- Step 4: Add assets such as ground and wall sprites
- Step 5: Select the background image that will be displayed in the game
- Steps 6 and 7: Add, edit or delete a game level
- Step 8: Give the game a tittle and a brief description
- Step 9: Download the generated game

Figure 1 (a) shows the first step of the generation process. Figure 1 (b) illustrates level 3 game play interface of a sample game built from the Mushroom picker game. It is worth noting that the generated games present players (students) with a pedagogical goal (learning outcome/ task) and an Integrated Development Environment (IDE) where students complete python code snippets. For instance, Table 1 illustrates the alignment of the Mushroom Picker game levels to pedagogy in CS1. For each level completed successfully, trophies and health points are awarded.

4 EXPERIMENTAL DESIGN

The proposed tool is evaluated next in an online experiment with educators in terms of its usability.

4.1 Participants

30 CS1 instructors from higher learning institutions from Kenya and South Africa participated in the evaluation. Kenya and South Africa were chosen mostly for convenience purposes. The subjects from South Africa were recruited from a list of programming lecturers created during the Southern African Computer Lecturer's Association (SACLA) conference in 2018. On the other hand, respondents from Kenya were recruited by snowball sampling through recommendations by instructors since the first author (a programming lecturer from Kenya) had already established a few contacts. The sample was stratified by age, highest education attained, institution, experience, gamers verses non gamers, and lastly use of GBL in teaching. 60% of the respondents considered themselves as gamers while only 20% had used games in teaching. Other demographic information is summarised in Table 2.

[1]https://programmingwithfun.net/

4.2 Procedure

Ethical clearance was sought from the University of Cape Town and other relevant institutions. Next, a pilot study was conducted with 5 instructors from the University of Cape Town to: (i) ensure that respondents understood the questionnaire terminology; (ii) reduce chances of bias due to leading questions; and (iii) ensure that the questionnaire could be completed in a reasonable amount of time. Once this was done, invitation emails were sent to potential participants for the main study. Willing participants signed up and completed a consent form. This was followed by sending a pretest survey. Upon receiving a completed pretest survey from each participant, a post test survey was then sent out. In addition, a link to the tool, a video tutorial and a task sheet were provided. The design approach adopted reduced chances of social desirability bias [20] and gave users a reasonable exposure to the tool (1 month). This reduced chances of variability in the usefulness dimension ratings [15]. Upon completing the posttest survey, interested participants entered into a draw for a chance to win a prize worth 500 South Africa Rand as compensation for their time and Internet data. In total, 10 prizes were available.

4.3 Data collection

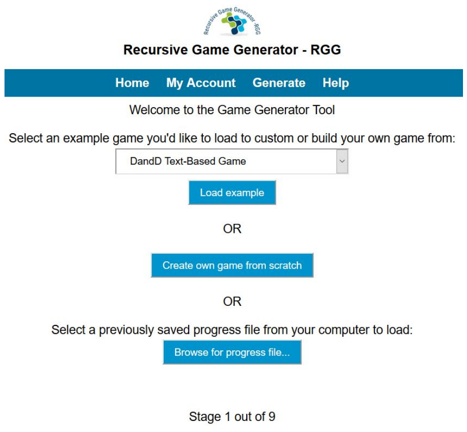
Quantitative UX data was collected through user self-reported data. Likert scales were used to collect both pretest and posttest study data. The standard Usefulness, Satisfaction, and Ease of use (USE) questionnaire [15] was adopted, given the recent evidence of its reliability and validity [13]. The pretest survey contained questions about demographics. On the other hand, the posttest survey had questions on the user task experience; RGG's usability and one last overall question on final comments about the tool. Data on the final user comments was collected through an open ended question. All responses were collected and stored using the LimeSurvey tool. This ensured reliability of the given answers as each participant only answered once using a given token.

4.4 Analysis and Presentation

Statistical Package for Social Scientists (SPSS) software was used to perform descriptive analysis on collected data. USE was then scored separately for each of the four dimensions (usefulness, ease of use, ease of learning, and user satisfaction) by calculating the percentages of respondents in each category of the Likert scale for each item in the dimension. Stacked bar charts were used to analyse and present the percentage of users who fall into each category or level [1]. These percentages were then compared across the categories or tasks. For the estimated time on task and perceived task success, histograms were used to present frequencies. Meanwhile, text responses from final user comments were categorised, coded and analysed thematically [34]. Emerging themes were discussed and agreed upon by the first and second authors. Only results from respondents who completed both the pretest and posttest questionnaires were included in the analysis.

5 RESULTS

For all the post task rating figures, the tasks are: Task 1 - creating an account and logging in, Task 2 - creating a custom game using a



Recursive Game Generator - RGG

Welcome to the Game Generator Tool

Select an example game you'd like to load to custom or build your own game from:

DandD Text-Based Game

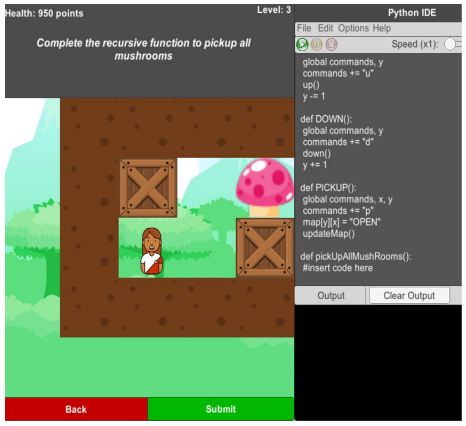
OR

OR

Select a previously saved progress file from your computer to load:

Stage 1 out of 9

(a) Game generation - step 1



Health: 950 points

Level: 3

Complete the recursive function to pickup all mushrooms

```
 global commands, y
 commands += "u"
 up()
 y -= 1

def DOWN():
 global commands, y
 commands += "d"
 down()
 y += 1

def PICKUP():
 global commands, x, y
 commands += "p"
 map[y][x] = "OPEN"
 updateMap()

def pickUpAllMushRooms():
 #insert code here
```

(b) Mushroom picker game Level 2 play interface

Figure 1: Examples of generated games

Table 1: Alignment of the Mushroom picker game levels to pedagogy in CS1

Level	Learning outcome/ task	Programming concept	Python code snippet

1	Complete the given code to pick up all mushrooms in the level	Function	`def pickUpAllMushrooms():`
			`#insert commands here`
			` #available commands:`
			`  RIGHT()`
			`  LEFT()`
			`  UP()`
			`  DOWN()`
			`  PICKUP()`

2	Complete the given code snippet by entering a stopping condition for the algorithm	Recursion - Stopping condition	`def pickUpAllMushrooms():`
			`#insert stopping condition below`
			` if():`
			`  return:`
			` RIGHT()`
			` PICKUP()`
			` RIGHT()`
			` pickUpAllMushrooms()`

3	Complete the recursive function to pick up all the mushrooms	Full Recursion	`def pickUpAllMushrooms():`
			`#insert code here`

given example, Task 3 - Customising the created game, and Task 4 - downloading the created game.

5.1 Task difficulty experience

To understand the level of difficulty users experienced when performing different tasks, participants were asked to rate the extent to which they agreed or disagreed with the statements on task easiness. Figure 2 shows a visualisation of the results in a stacked bar chart.

Overall, the findings suggest that most users found the tasks easy to perform with the RGG tool with creating user account/ logging in (94%) and downloading the created games (91%) being ranked as the easiest. This was followed by creating a custom game using a given example (74%) and lastly customising the game (71%). In total, another 20% found creating a game using an example difficult.

5.2 Task success

To evaluate the effectiveness of the tool in supporting educators when using it to create games to teach programming, we asked respondents to perform the four tasks and rate their levels of success. We measured task success on three levels (complete success, partial success and failure). Complete success and partial success were further broken down into success with or without assistance. On the other hand, failure was further categorised into two: (i) thought the task was easy but it was not; and (ii) failure - gave up. These levels were clearly explained to participants before using the tool [1]. Figure 3 is a visualisation of the results - where 'failure 2'stands for failure thought the task was easy but it was not, 'success 1'stands for partial success with assistance, 'success 2'stands for partial success without assistance, 'success 3'stands for complete success with assistance and 'success 4'stands for complete success without assistance. The findings suggest that participants were successful in most tasks. Considering the sum of participants who successfully completed the tasks with or without assistance, 27 (90%) successfully created a user account and logged in. This was followed by downloading the created game 25 (83%), creating a game using an example 23 (77%), and lastly customising the created

Table 2: Demographic information

Variable	**Category**	**Freq**	**Percentage %**
Age	25 - 30 years	2	6
	31 -35 years	5	16
	36 -40 years	8	27
	41 -45 years	11	38
	46 -50 years	3	10
	Above 50 years	1	3
Education	Bachelor degree	2	6
	Masters degree	20	67
	PhD. or higher	8	27
Institution	High school	2	6
	Tertiary college	4	14
	University	24	80
Experience	Less than 5 years	2	6
	5 -10 years	12	40
	11 -15 years	11	38
	16 -20 years	3	10
	Over 20 years	2	6

game 20 (66%). Two participants (6%) reported failure in tasks 2 and task 3. Finally, no failure was reported in task 1 and task 4.

5.3 Estimated time on tasks

We asked participants to give an estimate of the time they spent while performing each of the four tasks to measure the tool's efficiency. This was done on a 5 point scale. Time ranges (discrete time intervals) were used, where 1 represented 10 to 13 minutes. 2 represented 7 to 10 minutes, 3 represented 5 to 7 minutes, 4 represented 3 to 5 minutes and 5 represented 0 to 3 minutes. Figure 4 is a visualisation of the spread of completion times by all users presented as frequencies of participants in each scale. Results suggest that 23 (77%) users spent 0-3 minutes in task 1 followed by 15 (50%) in task 4 in the same time category. Comparing task 2 and task 3 under the category of users who estimated spending 10 - 13 minutes, task 2 had 5 (17%) of the respondents while task3 had 4 (13%). No participant spent more than 7 minutes in task 4.

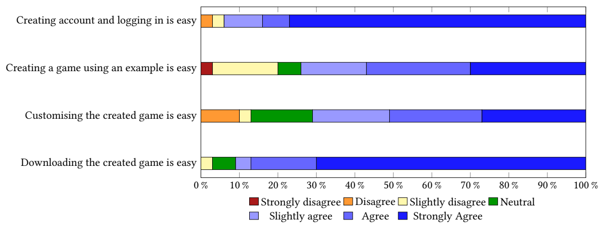


Figure 2: Task Difficulty Rating

5.4 Usability

To further evaluate the usability of the RGG tool, respondents answered the USE questionnaire [15], which has four dimensions:

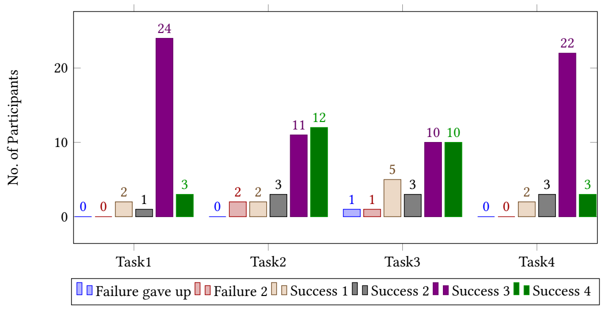


Figure 3: Task Success Rating

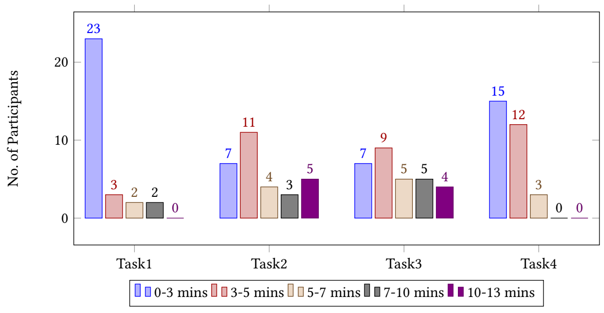


Figure 4: Estimated Time on Task Rating

Usefulness, Ease of use, Ease of learning and Satisfaction. Participants' ratings of the tool's usability on these dimensions were scored from 1 for strongly disagree to 7 for strongly agree.

5.4.1 *Usefulness.*

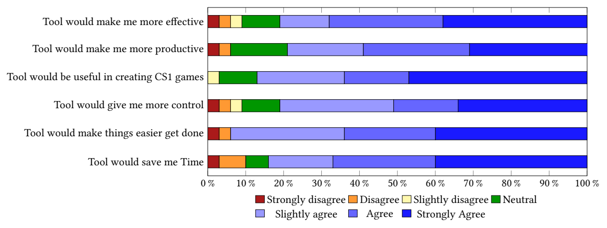
Results obtained suggest that most CS1 lecturers surveyed answered affirming the usefulness of the RGG tool. As can be seen in the stacked bar chart in Figure 5a, overall, 87% agreed that the tool would be useful for helping programming instructors to create games to teach the recursion topic in CSI. Another 94% of the participants agreed that the tool would make the things programming instructors want to accomplish with the tool easier to get done. On the other hand, 84% said that the tool would save them time when creating games to teach programming. Lastly, 81% reported that the RGG tool would either make instructors more productive in their teaching, more effective or give them control over their teaching activities. However, some 10% of the users found the tool not useful in saving time, suggesting possible improvements in future. These results demonstrate the pedagogical promise of the RGG game generator tool in teaching programming in higher education.

5.4.2 *Ease of Use.*

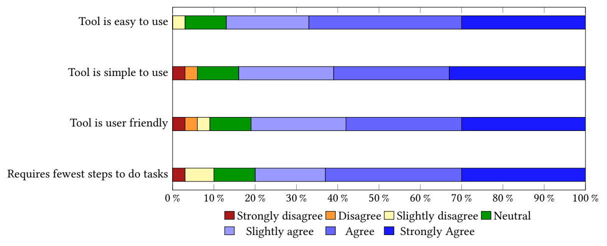
Figure 5b presents the subjective rating of the tool's ease of use by participants. Clearly, most participants found the tool easy to use (87%), simple to use (81%), usable (80%) and requiring the fewest steps to accomplish tasks (74%).

5.4.3 *Ease of Learning.*

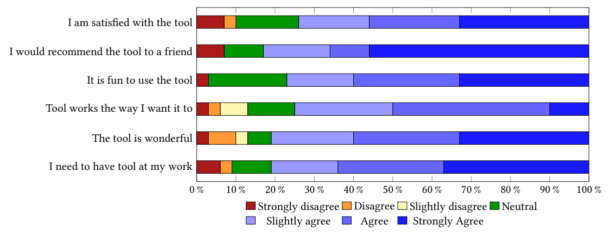
When asked to evaluate the usability of the tool on the learnability dimension, results suggest that, overall, participants found RGG easy to learn. In total, about 80% of the respondents said that they either learnt the tool quickly or that they could easily remember



(a) Tool Usefulness Rating



(b) Ease Of Use Rating



(c) Satisfaction Rating

Figure 5: Tool Usability Ratings

how to use it. Another 77% reported that it was easy to learn how to use it. Generally, these results suggest learnability of the tool. Among the participants who disagreed, 13% reported that it was neither quick nor easy to learn to use the tool.

5.4.4 *User Satisfaction.*

We asked the participants the extent to which they agreed or disagreed that they were satisfied with the tool based on the satisfaction items in the USE questionnaire. Figure 5c presents a summary of responses as a percentage. In total, results obtained show that 83% of programming instructors agree that they would recommend the RGG generator tool to a friend with 56% strongly agreeing. Another 81% agreed that they both needed to have the tool at their work place and that it was wonderful. On the other hand, 77% felt that the tool was fun to use. Lastly, approximately 75% of the respondents were either satisfied with RGG or thought it worked the way they wanted it to.

5.5 Final user comments

Lastly, we asked respondents to give one final comment about the RGG tool to gain more insight about their impressions. Respondents were of the opinion that RGG was a useful and noble tool for supporting teaching programming in higher education (80%). In addition, a majority (75%) strongly recommended it's adoption in teaching programming in higher education. Below are some direct exemplars:

- "Its a useful learning tool to us lecturers and students."
- "I strongly recommend it for adoption in teaching Recursion in Computer Programming."

6 DISCUSSION

Overall, findings from this study's usability evaluation with educators agree with those of previous similar work [30]. The finding that the prototype can generate games that can allow students to practice coding is in line with the design principle for programming games [5]. For games generated from the Mushroom picker and Runner top down maze examples - upon completing the code snippet, students can visualise the execution of their code/ algorithm through a character moving recursively as recommended by Chaffin et al. [5]. Additionally, the games created from the mushroom picker example use simple commands (LEFT(), RIGHT(), UP(), DOWN()) for novices in sync with the ALICE programming game [6]. The game generator prototype not only extends these design principles, it also authors customisable games inline with recommendations in [18]. One limitation of the current study is the missing feedback from students. However, the primary users of the proposed tool are CS1 educators who interact directly with it. Students are merely secondary users who are affected by the capability of the primary users to carry out tasks with the tool [16]. None the less, a follow up study will attempt to address this issue. Consequently, by testing the idea of a game generator tool using a prototype and evaluating it with instructors, positive findings on usability and general comments potentially demonstrate that the tool could be suitable for adoption by CS1 instructors to support teaching novices. We therefore argue that the evidence is valid and useful for advancing GBL in the context of Africa but could also be tested globally by the wider CSE research community to further GBL research.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we presented findings from design and usability evaluation of a prototype game generator tool called RGG. We evaluated the prototype with 30 CS1 instructors from Kenya and South Africa who answered a survey on the usability. Results demonstrate that the prototype is effective, efficient, useful, and easy to use and learn. Additionally, instructors are satisfied with the prototype and highly recommend adoption of a game generator tool idea for teaching programming. Future research direction will involve testing the prototype with an instructor and CS1 students in a semester course offering to provide preliminary insights on how it would practically work in a learning set up.

8 ACKNOWLEDGEMENTS

This research was financially supported by Hasso Plattner Institute for Digital Engineering HPI, the National Research Foundation of South Africa (Grant numbers: 85470 and 88209) and University of Cape Town. The authors acknowledge that opinions, findings and conclusions or recommendations expressed in this publication are that of the authors, and that the NRF accepts no liability whatsoever in this regard.

REFERENCES

[1] William Albert and Thomas Tullis. 2013. *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes.

[2] Jecton Tocho Anyango and Hussein Suleman. 2018. Teaching Programming in Kenya and South Africa: What is difficult and is it universal?. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research*. 1–2.

[3] Johan Baldeón, Anna Puig, Inmaculada Rodríguez, Cristian Muriel, and Leandro Zardain. 2017. A Conceptual Model for Educational Game Authoring: A Showcase in Math Games. In *Design, User Experience, and Usability: Designing Pleasurable Experiences*, Aaron Marcus and Wentao Wang (Eds.). Springer International Publishing, Cham, 347–361.

[4] Yosra Bouzid, Mohamed Ali Khenissi, and Mohamed Jemni. 2015. Designing a game generator as an educational technology for the deaf learners. In *Information & Communication Technology and Accessibility (ICTA), 2015 5th International Conference on*. IEEE, 1–6.

[5] A Chaffin, Katelyn Doran, Drew Hicks, and Tiffany Barnes. 2009. Experimental evaluation of teaching recursion in a video game. In *In Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games, Sandbox '09*. ACM, 79–86.

[6] Stephen Cooper, Wanda Dann, and Randy Pausch. 2000. Alice: a 3-D tool for introductory programming concepts. *Journal of computing sciences in colleges* 15, 5 (2000), 107–116.

[7] Matteo Gaeta, Vincenzo Loia, Giuseppina Rita Mangione, Francesco Orciuoli, Pierluigi Ritrovato, and Saverio Salerno. 2014. A methodology and an authoring tool for creating Complex Learning Objects to support interactive storytelling. *Computers in Human Behavior* 31 (2014), 620–637.

[8] Roslina Ibrahim and Azizah Jaafar. 2009. Educational games (EG) design framework: Combination of game design, pedagogy and content modeling. In *2009 International Conference on Electrical Engineering and Informatics*, Vol. 1. IEEE, 293–298.

[9] Chris Johnson, Monica McGill, Durell Bouchard, Michael K. Bradshaw, Víctor A. Bucheli, Laurence D. Merkle, Michael James Scott, Z. Sweedyk, J. Ángel Velázquez-Iturbide, Zhiping Xiao, and Ming Zhang. 2016. Game Development for Computer Science Education. In *Proceedings of the 2016 ITiCSE Working Group Reports* (Arequipa, Peru) *(ITiCSE '16)*. Association for Computing Machinery, New York, NY, USA, 23–44. https://doi.org/10.1145/3024906.3024908

[10] Aous Karoui, Iza Marfisi-Schottman, and Sébastien George. 2017. A Nested Design Approach for Mobile Learning Games. In *Proceedings of the 16th World Conference on Mobile and Contextual Learning* (Larnaca, Cyprus) *(mLearn 2017)*. Association for Computing Machinery, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.1145/3136907.3136923

[11] Mohamed Ali Khenissi, Fathi Essalmi, and Mohamed Jemni. 2015. Comparison between serious games and learning version of existing games. *Procedia-Social and Behavioral Sciences* 191 (2015), 487–494.

[12] Devorah Kletenik and Deborah Sturm. 2018. Game Development with a Serious Focus *(SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 652–657. https://doi.org/10.1145/3159450.3159588

[13] Gao M Kortum P and Oswald F. 2018. Psychometric Evaluation of the USE (Usefulness, Satisfaction, and Ease of use) Questionnaire for Reliability and Validity. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. 1414–1418.

[14] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A study of the difficulties of novice programmers. *Acm sigcse bulletin* 37, 3 (2005), 14–18.

[15] Arnold M Lund. 2001. Measuring usability with the use questionnaire12. *Usability interface* 8, 2 (2001), 3–6.

[16] Martin Maguire. 2001. Context of use within usability activities. *International Journal of Human-Computer Studies* 55, 4 (2001), 453–483.

[17] Sohail Iqbal Malik and Jo Coldwell-Neilson. 2017. A model for teaching an introductory programming course using ADRI. *Education and Information Technologies* 22, 3 (2017), 1089–1120.

[18] Eugenio J Marchiori, Javier Torrente, Ángel del Blanco, Pablo Moreno-Ger, Pilar Sancho, and Baltasar Fernández-Manjón. 2012. A narrative metaphor to facilitate educational game authoring. *Computers & Education* 58, 1 (2012), 590–599.

[19] Florian Mehm. 2010. Authoring Serious Games. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games* (Monterey, California) *(FDG '10)*. Association for Computing Machinery, New York, NY, USA, 271–273. https://doi.org/10.1145/1822348.1822390

[20] Clive Nancarrow and Ian Brace. 2000. Saying the "right thing": Coping with social desirability bias in marketing research. *Bristol Business School Teaching and Research Review* 3, 11 (2000), 1–11.

[21] Jakob Nielsen. 1994. *Usability engineering*. Morgan Kaufmann.

[22] Joseph C Osborn, Melanie Dickinson, Barrett Anderson, Adam Summerville, Jill Denner, David Torres, Noah Wardrip-Fruin, and Michael Mateas. 2019. Is Your Game Generator Working? Evaluating Gemini, an Intentional Generator. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 15. 59–65.

[23] José Carlos Paiva, Ricardo Queirós, José Paulo Leal, and Jakub Swacha. 2020. FGPE AuthorKit – A Tool for Authoring Gamified Programming Educational Content. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE '20)*. Association for Computing Machinery, New York, NY, USA, 564. https://doi.org/10.1145/3341525.3393978

[24] Kaśka Porayska-Pomsta, Keith Anderson, Sara Bernardini, Karen Guldberg, Tim Smith, Lila Kossivaki, Scott Hodgins, and Ian Lowe. 2013. Building an intelligent, authorable serious game for autistic children and their carers. In *International Conference on Advances in Computer Entertainment Technology*. Springer, 456–475.

[25] V. M. Pérez-Colado, I. J. Pérez-Colado, M. Freire-Morán, I. Martínez-Ortiz, and B. Fernández-Manjón. 2019. uAdventure: Simplifying Narrative Serious Games Development. In *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, Vol. 2161-377X. 119–123.

[26] Arturo Barajas Saavedra, Francisco J. Álvarez Rodríguez, Jaime Muñoz Arteaga, René Santaolaya Salgado, and César A. Collazos Ordoñez. 2014. A Serious Game Development Process Using Competency Approach: Case Study: Elementary School Math. In *Proceedings of the XV International Conference on Human Computer Interaction* (Puerto de la Cruz, Tenerife, Spain) *(Interacci&#243;n '14)*. ACM, New York, NY, USA, Article 99, 9 pages. https://doi.org/10.1145/2662253.2662352

[27] Jeff Sauro and James R Lewis. 2016. *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann.

[28] Alexis Sepchat, Nicolas Monmarché, Mohamed Slimane, and Dominique Archambault. 2006. Semi automatic generator of tactile video games for visually impaired children. In *International Conference on Computers for Handicapped Persons*. Springer, 372–379.

[29] Esther Shein. 2015. Python for beginners.

[30] Javier Torrente, Ángel Del Blanco, Eugenio J Marchiori, Pablo Moreno-Ger, and Baltasar Fernández-Manjón. 2010. < e-Adventure>: Introducing educational games in the learning process. In *IEEE EDUCON 2010 Conference*. IEEE, 1121–1126.

[31] Javier Torrente, Pablo Moreno-Ger, Baltasar Fernández-Manjón, and José Luis Sierra. 2008. Instructor-oriented authoring tools for educational videogames. In *2008 Eighth IEEE International Conference on Advanced Learning Technologies*. IEEE, 516–518.

[32] Javier Torrente, Ángel Serrano-Laguna, Conor Fisk, Breid O'Brien, Wanda Alesky, Baltasar Fernández-Manjón, and Patty Kostkova. 2015. Introducing Mokap: A Novel Approach to Creating Serious Games. In *Proceedings of the 5th International Conference on Digital Health 2015* (Florence, Italy) *(DH '15)*. ACM, New York, NY, USA, 17–24. https://doi.org/10.1145/2750511.2750529

[33] Irene Visscher-Voerman and Kent L Gustafson. 2004. Paradigms in the theory and practice of education and training design. *Educational Technology Research and Development* 52, 2 (2004), 69–89.

[34] Rebekah Willson. 2019. Analysing Qualitative Data: You Asked Them, Now What to Do With What They Said. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval* (Glasgow, Scotland UK) *(CHIIR '19)*. ACM, New York, NY, USA, 385–387. https://doi.org/10.1145/3295750.3298964