Case Studies on the use of Storyboarding by Novice Programmers

Ally Limke
Alexandra Milliken
anlimke@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Veronica Cateté
Isabella Gransbury
Amy Isvik
North Carolina State University
Raleigh, NC, USA

Thomas Price
Chris Martens
Tiffany Barnes
tmbarnes@ncsu.edu
North Carolina State University
Raleigh, NC, USA

ABSTRACT

Our researchers seek to support students in building block-based programming projects that are motivating and engaging as well as valuable practice in learning to code. A difficult part of the programming process is planning. In this research, we explore how novice programmers used a custom-built planning tool, PlanIT, contrasted against how they used storyboarding when planning games. In a three-part study, we engaged novices in planning and programming three games: a maze game, a break-out game, and a mashup of the two. In a set of five case studies, we show how five pairs of students approached the planning and programming of these three games, illustrating that students felt more creative when storyboarding rather than using PlanIT. We end with a discussion on the implications of this work for designing supports for novices to plan open-ended projects.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; • **Human-centered computing** → **User studies**.

KEYWORDS

block-based programs, open-ended projects, planning, storyboards

ACM Reference Format:
Ally Limke, Alexandra Milliken, Veronica Cateté, Isabella Gransbury, Amy Isvik, Thomas Price, Chris Martens, and Tiffany Barnes. 2022. Case Studies on the use of Storyboarding by Novice Programmers. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol 1 (ITiCSE 2022), July 8–13, 2022, Dublin, Ireland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3502718.3524749

1 INTRODUCTION

Curricula using block-based programming languages like Snap*!* [10] have engaged students in open-ended, creative programming [6, 8]. These approaches to teaching can help broaden participation in computing by connecting material with students' personal interests [14, 19]. While open-ended programming has many benefits, prior work shows students struggle with aspects of the design process like communicating ideas and articulating algorithms [17]. As open-ended programming can motivate students to participate in computing, it is important to support students in their designs.

Existing design tools are primarily text based [1, 11, 13]. In visual, interactive block-based environments, it may be helpful for students to create visual compositions using the graphics from their programs. Therefore, we introduce simple digital storyboarding as a game design technique for block-based programming. We engaged novices in planning and programming three games: a maze, break-out, and a mash-up of the two. Through case studies, we show how pairs of students planned and programmed these games, finding five planning themes: Mechanic Illustrators, Idea Explorers, Confidence Builders, Artists, and Confident Programmers. We focused on answering the overarching question: How do novice programmers perceive the impact of planning using storyboarding or a planning tool on their experiences making games in a block-based language? These case studies provide insights into how different planning tools may engage students with varying levels of confidence and experience to make complex, interactive programs.

2 RELATED WORKS

Prior work shows that students struggle with ideating, creating prototypes, and articulating algorithms for open-ended projects [17]. Additionally, students have trouble describing original game mechanics [2]. Failure to properly plan for programming tasks may result in poorly-organized code [12] and keep students from achieving their goals. Many students who gain programming proficiency in block programming have duplicate code and long scripts [16].

Because planning helps students program open-ended tasks, multiple tools have been developed to support the process. Simple paper tools for game design in block-based languages include Design Notebooks [17] and mechanic planning [2]. Other digital tools have been developed and integrated into programming environments, like a Java planning tutor [11], a UML class diagram plugin for Eclipse [1], and PlanIT for Snap*!* [13].

Whereas the tools above are primarily text-based, storyboarding is a visual approach that remains largely unexplored as a design tool for block-based games. Storyboarding is a common technique used by designers and HCI specialists to illustrate system interfaces [18]. Storyboarding has been used by K-12 students for designing video games [4, 17] and games for augmented reality storytelling [7]. Paper storyboards were used to help middle school students plan for animations in Scratch [5, 15]. However, none of these studies analyze how students used the storyboards or detail their experiences. This is an pertinent gap in the literature as it is important

ITiCSE 2022, July 8–13, 2022, Dublin, Ireland

ACM ISBN 978-1-4503-9201-3/22/07…$15.00
https://doi.org/10.1145/3502718.3524749

to know if storyboarding is an effective tool for planning and how storyboarding compares to existing planning tools.

3 PLANIT & STORYBOARDING TEMPLATES

PlanIT is a text-based planning tool integrated into the Snap*!* programming environment[13] that allows students to add descriptions to their games and define a To Do list. PlanIT also supports students in defining events occurring between their sprites and variables. The planned sprites and variables are automatically added to Snap*!* so that students make progress on their final programs as they plan.

We provided storyboard templates in Google Slides, allowing for concurrent editing. Each template included a space to construct a scene, add a title, and write a description. There were three templates: a title slide, a single scene (see Figure 2), and a "When/Then" slide which included two scenes. The single scene slide also included a space labeled "Trigger" to help students think about what causes a scene to take place.

4 CONTEXT

We conducted a study with 36 high school students during week five of a six week virtual summer computing internship led by a graduate student and faculty member at a large public university [3]. Students were recruited from across the country and had a variety of school backgrounds. During the internship, students acted as code support for K-12 teachers creating block-based lessons for their classroom and were familiar with coding in block-based languages before the study. Of the 36 interns, the demographics for the 10 consenting students that we focus on are presented in Table 1 with a self-assessment of their programming confidence on a Likert scale: no programming skills (1), little (2), some (3), and very strong (4). All student names have been replaced with pseudonyms.

5 METHODS

5.1 Research Questions

To better understand how storyboarding helps students plan novel programs, our research questions are: **RQ 1:** How do novice programmers perceive the impact of storyboarding on their experiences planning and programming games in a block-based language? **RQ 2:** How do novice programmers perceive their experiences with storyboards in comparison to the planning tool, PlanIT?

We expect students will find the storyboarding templates helpful for collaboration as they allow for visual communication of ideas. We predict students will structure their planning time by first using the storyboarding templates and then PlanIT. Storyboarding allows students to construct general game ideas, whereas PlanIT assists students in thinking about code mechanics. Finally, we expect students will perceive the storyboarding templates to support their creativity through allowing students to iterate on their designs and create programs that are meaningful to them.

5.2 Study Design

As storyboarding is often used for communication, we assigned students to pairs. All students participated in a PlanIT tutorial. The study was conducted in three sessions over two days. On Day 1, students engaged in one session. On Day 2, students participated in two separate sessions. Each session had the following progression: Students were introduced to the game genre and explored example games in the Scratch library. (15 min) Students planned for a game using assigned planning tools in Zoom breakout rooms. (30 min) Students pair programmed the games in Snap*!*. (90 min) Students participated in four-person focus groups where they presented their game and answered questions about their experiences planning and programming. (15 min) Students took a post-survey to further describe their experiences with the planning tools and experiences pair programming. (10 min) We collected students' code traces, coding artifacts, post-surveys, and focus group transcripts.

Since we are interested in how PlanIT compares to storyboarding, we partitioned students into two conditions: Students in Condition A could use both PlanIT and storyboarding slides for planning, while the students in Condition B could only use PlanIT. Students were only introduced to the tools they were allowed to use.

Each session featured a different game genre: a maze game, a breakout game, and a mashup of their first two games. We included the mashup style game to explore students' experiences representing novel ideas in addition to planning for the existing games. Each planning and programming period was self-directed. For storyboarding, we instructed students to duplicate slides, delete slides, and use the templates in ways that fit their needs. We provided pairs in both conditions with sample sprites which were intended to give them a starting point and prevent them from spending their planning time searching for sprites online.

6 CASE STUDIES

The following case studies detail the participants' planning and programming journeys. We focus our case studies on groups in Condition A to illustrate student preference and the affordances of each tool. We conclude with one case study from Condition B to illustrate how planning in PlanIT contrasts to planning with both tools. While all students in Condition A participated in all sessions, many participants in Condition B missed some planning or programming sessions. The case study group from Condition B we

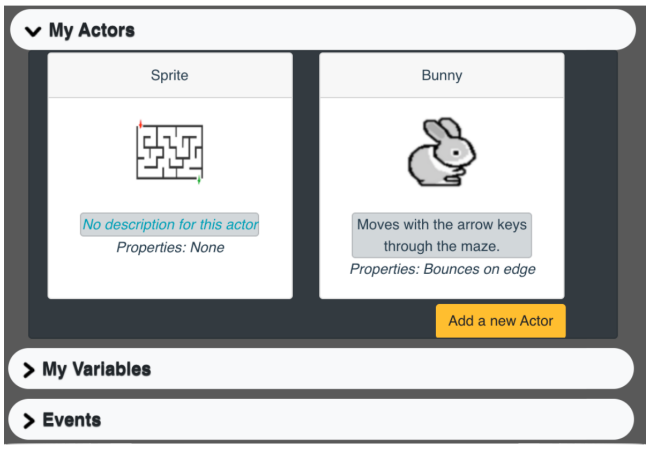

Figure 1: PlanIT, Planning Tool

Table 1: Participants' Demographic Data

Team	Group	Gender	Ethnicity	Programming confidence
Duck & Fish	A	F, F	Asian / Pacific Islander, Black	3, 4
Cat & Wasp	A	F, M	Southeast Asian / Indian, Hispanic / Latin(x)	4, 4
Bat & Koala	A	F, F	Southeast Asian / Indian, Asian / Pacific Islander	2, 3
Deer & Zebra	A	F, F	Black, Black	3, 3
Octopus & Robin	B	F, F	White, Southeast Asian / Indian	3, 4

chose to focus on attended all sessions. Those in Condition B who were not discussed in our case studies have similar programming experiences and demographics as others in the study. We use each group's code traces, survey responses, planning artifacts, coding artifacts, and focus group transcripts to construct each case study.

To provide further context to the case studies, two of the authors hand-analyzed the plans and final code to determine the number of features students planned for and how many they implemented in Table 2. When calculating the total, only features that were beyond the scope of the game instructions were counted (e.g. levels, GUI).

Table 2: Counts of features programmed and planned.

Team	Group	Features planned			Features programmed		
		G1	*G2*	*G3*	*G1*	*G2*	*G3*
Duck & Fish	A	9	5	7	9	4	3
Cat & Wasp	A	4	2	7	6	8	10
Bat & Koala	A	5	0	8	5	0	8
Deer & Zebra	A	9	1	10	8	3	16
Octopus & Robin	B	5	1	5	8	4	7

6.1 Duck & Fish: Mechanic Illustrators

Duck and Fish, the *Mechanic Illustrators*, show how storyboards can aid in collaboration and in ideating game mechanics. This case suggests that students find storyboarding of greater value when they are planning for more complex concepts.

Maze Duck and Fish planned thoroughly for their maze game, constructing seven scenes, writing pseudocode, and defining a win condition. Whereas some groups focused on creating a unique theme for their game, this group appreciated that the slide deck provided sprite costumes so they could "focus on building [their] ideas" for their game mechanics. Fish arrived late to the planning session. In the post-session interview, both group members emphasized that the storyboards were "really helpful to group programming," allowing Fish to quickly understand Duck's plans.

Breakout This group created detailed plans for a basic breakout game with the addition of a "lives" variable. The storyboards were sequential with starting and ending slides, arrows for denoting movement, and descriptions of conditionals and variables. After session one, Duck shared that storyboarding felt unhelpful "for coding activities that can be finished in less than two hours." However, Duck's perspective changed after programming the breakout game. Duck shared that storyboarding helped them "compartmentalize [their] thinking" and "visualize the if/then conditions". It may be that for games which require more advanced Snap*!* concepts (e.g. cloning), storyboarding feels more worthwhile.

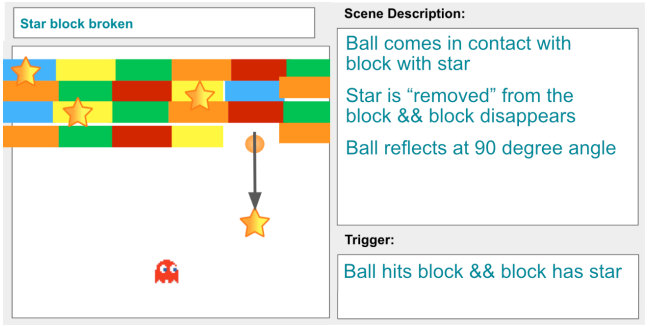

Figure 2: Duck & Fish's Mashup Slide

Mashup This group used their planning time productively focusing only on the parts of their mashup game they had not programmed in past sessions. Rather than rebuilding their storyboards, this group copied and pasted their slides from Breakout and added three slides to describe a new feature for their game, including the slide in Figure 2. Similar to Cat and Wasp, who used the storyboards to experiment with ideas rather than making detailed plans, this group only planned for what would be useful for them: the new feature. This indicates that storyboards are useful as a tool to ideate new mechanics for their programs rather than serve as busy work. Duck and Fish did not use PlanIT in the planning sessions. Duck shared that, after completing the storyboards, the group "found the PlanIT feature unnecessary" and "redundant" with their storyboarding plans. This indicates that this group may have found value in using PlanIT if the storyboarding slides were unavailable, and simply preferred to use the storyboarding slides.

6.2 Cat & Wasp: Idea Explorers

Cat and Wasp, the *Idea Explorers*, illustrate that storyboards allow students to explore game ideas at a low time-cost. They also reveal that while storyboarding is the preferred planning tool among the groups, PlanIT may be better at supporting textual planning. Finally, this case reveals that students with more programming experience may find the storyboards unhelpful for simple games.

Maze Cat and Wasp chose to plan only with storyboards. While the group's maze plans were thorough, Wasp shared that they were not helpful, but for games with more mechanics, it would be useful to "be able to refer back" them. Wasp and Cat had the highest confidence programming in Snap*!* compared to others in Condition

A. This may explain why Wasp felt the storyboards were unhelpful when the other groups found them both helpful and enjoyable.

Breakout Cat and Wasp used the storyboards as an exploratory tool to plan for their soccer themed breakout game. Wasp emphasized that the storyboards let their team explore ideas without investing time into programming them. These storyboards were less detailed, only describing the interaction between the ball and bricks. Although their storyboards looked incomplete, it seems the group only used the storyboards for parts of the program that they were unsure about, which is a productive use of the planning time. The group's final game had more features than they had planned for. Wasp explained that the group did not use PlanIT because they "had everything [they] needed already planned from storyboarding."

Mashup The group used both PlanIT and the storyboards to plan for their game. PlanIT was used to describe user controls and game rules, whereas the storyboards had little text and were mostly visual. The group may have used both tools because the mashup game required them to think of an original idea, which they described was "really hard".

6.3 Bat & Koala: Confidence Builders

Bat and Koala, the *Confidence Builders*, began the study unsure of their programming abilities. Bat was the only student to report having very little programming skills despite having experience in multiple languages. Koala was the only student that reported feeling unconfident in their ability to program an existing game in Snap*!*. This case illustrates that even students who lack confidence and struggle with storyboarding may feel proud of their storyboards and enjoy storyboarding. This case describes how storyboarding can build students' confidence in their ability to program games.

Maze Koala and Bat struggled with representing certain mechanics on their storyboards. For example, Bat shared that "making clear how the door would open" was the hardest part of planning because it could be interpreted differently than their intent. After each session, this group shared that they struggled to represent their intent visually. Providing students with examples of other storyboards may help them understand how others represented similar mechanics. Although Bat and Koala programmed a complete game, the group shared that they were most proud of creating their storyboards. Because planning felt difficult for this group, overcoming the challenge may have felt more rewarding to them than programming the game.

Breakout This group's lack of confidence is reflected in the language used in their breakout storyboards. They used terms that expressed uncertainty in their design or programming abilities; for instance,"a variable that *could* change is the number of bricks that disappeared". This uncertainty may have been a result of the increased difficulty of the breakout game and may be why this sentiment did not appear in their maze slides. Bat and Koala's hesitancy is also reflected in their plans for the breakout game. They did not plan or program any extra features.

Mashup This group expressed worry in their ability to complete the mashup task. Bat said that their first reaction was "What? How can we combine both of these to make something in the time that we have? [...] We weren't sure what to do." However, Bat and Koala shared that the process of storyboarding allowed them to experiment with ideas for their game and create an implementable plan. In fact, Bat and Koala implemented all the features that they planned for in their mashup. In the post-survey, we saw that Bat's programming self-assessment increased. This may be a result of successfully programming a game that once seemed daunting.

Koala's confidence in their ability to create an existing game in Snap*!* increased in the post-survey. In session one, Bat and Koala used PlanIT to describe their game and add sprites. However, in their mashup Bat and Koala did not use PlanIT for any aspect of planning. As their confidence in their programming skills increased, they likely felt less need to create detailed plans. Based on prior research, undergraduate students reported feeling "bad at programming" if they needed to plan before writing code [9]. Therefore, if the students felt more confident in their programming abilities, they may have found it unnecessary to use multiple planning techniques.

6.4 Deer & Zebra: The Artists

Deer and Zebra, the *Artists*, reveal that storyboarding can help inspire students and make them feel supported creatively by allowing them to quickly express their ideas. However, this case also illustrates that the visual aspect of storyboarding may distract students from planning more complex mechanics, instead encouraging them to focus on finding interesting graphics.

Maze Deer and Zebra chose not to use PlanIT and planned only using the storyboarding templates. This is similar to other groups in Condition A who also did not use PlanIT for at least one of their games indicating a preference for storyboarding over PlanIT.

After completing their game, Deer and Zebra shared that storyboarding gave them inspiration and removed the blank page syndrome that they often feel when starting a project in Snap!. Deer and Zebra shared that they felt most creative while storyboarding rather than when they were programming. This is interesting as much of the text that accompanied their storyboard visualizations were written with Snap*!* coding terms like "Broadcast" and conditional statements showing that storyboards allow students to feel creative while using their computational thinking skills.

Although the group enjoyed and felt they benefited from storyboarding, they also expressed struggling with the process. The group shared that they did not know how to organize their slides and had trouble determining "what game mechanics are one or two different scenes". This focus on communicating mechanics rather than a sequential story is reflected in the group's storyboards. For all three of their games, the storyboards were not used sequentially, but instead to isolate a mechanic in their game.

Breakout Deer and Zebra's breakout game had fewer features than our instructions suggested. While their game mechanics were simple, their visuals were not, creating a unique candy theme. Zebra shared that they most enjoyed adding the sprites to the storyboards and "making it look nice." Thus, students in some cases may be focused on using the storyboards to plan for a game theme rather than ideating ambitious game mechanics that challenge them. The lack of game mechanics in their plans is reflected in their final code. The group's breakout game contained only 52 blocks, which was the second-least number of blocks used by any group.

Mashup Unlike their breakout game, Deer and Zebra's mashup was very ambitious, incorporating collectible items and a shooting

mechanism into their maze. The challenge of creating an original game may have taken the group's focus away from planning for visual attributes and instead shifted it toward game function.

The students programmed more than they planned for, implementing six additional features like a timer and an ending message. The success this group experienced, when they did not spend their time searching for sprite costumes, reveals a need for a limit on the time a group may spend looking for sprites or supports to make the process faster. A themed sprite library with helpful search features may resolve the issue in future storyboard tools. We can not dismiss interest in searching for sprite images as unproductive as it may be that this group's excitement about their graphics helped them stay motivated to complete their games.

6.5 Octopus & Robin: Confident Programmers

This group had one of the most confident programmers, Robin, who evaluated their programming skills as very strong. This case shows that confident students may feel that planning is unhelpful when designing games in Snap*!*, but that visuals may make planning feel more purposeful. This group differs from the other case studies in that they did not have access to the storyboards.

Maze Octopus shared that it was hard to translate their plans into words because they couldn't "visualize the stuff that would have to be done later in the program." This contrasts with Bat and Koala, who relied heavily on text and struggled to use visual storyboarding scenes. Perhaps the ability within storyboards to do both visual and textual planning gives students more options to express their ideas when one type of planning feels difficult.

Breakout This group used PlanIT to add actors and events that describe user input and sprite interactions. After programming the game, Robin shared that the planning did not feel necessary, as the group was "writing things down just to write them" and that they "didn't refer back to it afterwards." Robin was the more confident programmer, so it is possible that they felt more comfortable starting programming without any planning.

Mashup In contrast to their first two games, Octopus and Robin's mashup game was ambitious and unique. They planned for a game in which a Pac-man sprite is launched from the bottom of the screen and moves upward into a maze-like structure.

Robin did not think planning "added to [their] coding experience." Some students view visual programming environments as inferior, believing they do not support complex coding tasks [20]. This may explain why this group's plans and maze and breakout games were not very ambitious. Planning may have felt purposeful to this group if they had a visual component available to help ideate their game plans. Octopus reiterated that it was hard to visualize their plans, stating that "imagining the event plans" was difficult.

7 DISCUSSION

Findings from our case studies to answer RQ1 can be sorted into three themes: storyboards support creative thinking, conditions for valuing the storyboarding process, and storyboarding struggles and solutions. We then address RQ2 and then discuss the limitations of the current study. We refer to each case study with the following abbreviations: Bat and Koala (BK), Deer and Zebra (DZ), Duck and Fish (DF), Cat and Wasp (CW), and Octopus and Robin (OR).

7.1 Storyboarding supports creativity

DZ and BK shared that they felt most creative while they were storyboarding rather than programming. The process of storyboarding may help support creativity because it allows students to experiment with their ideas and quickly make changes. CW shared that storyboarding allowed their team to revise their plans when they had new ideas. The ability to do iterative design before starting programming may have helped students feel more comfortable starting their projects and prouder of their results. DZ shared that the storyboards inspired them and removed the daunting feeling they get when they stare at the blank scripting space in Snap*!*. Similarly, when BK were unsure in their ability to complete the mashup game, they said that storyboarding helped them experiment with their ideas and create implementable plans. BK shared that they felt most proud of completing their storyboards, indicating a sense of ownership over the final ideas.

Design Implications As seen above, planning is a non-sequential iterative process for novice programmers. Future planning tools should allow students the flexibility to alter their designs quickly. It seems that to support creativity, planning should have a visual aspect, not just textual. When BK and DZ shared that they felt most creative storyboarding, they mentioned that the creativity was due to the ability to see how their game would look when completed. OR, the team in Condition B, shared that they had a hard time planning textually because they could not visualize their program. Additional visual planning may easily be added to PlanIT; when students define events (e.g. interactions between sprites, user input) the system may prompt them to create a corresponding storyboard.

7.2 Conditions for valuing storyboarding

Wasp and Duck initially had negative attitudes towards storyboarding, but over the progression of the three sessions they started to feel that the storyboards were useful to them. They valued storyboarding when 1.) the games were challenging and 2.) when they needed to communicate their ideas to their partner.

Planning for challenges When creating storyboards for their mashups, all groups planned for new game features rather than for the functionality they programmed in previous sessions showing that these students view the storyboarding slides as a planning *tool* rather than as a *product*. This is seen in CW's breakout and mashup storyboards; they planned only for features that they found useful, rather than creating a complete plan. Therefore, if we want students to benefit from and utilize storyboarding, the complexity of the games they plan for must match or exceed their current programming abilities. DF and CW, found storyboards more helpful as the games increased in difficulty over the three sessions.

Storyboarding for collaboration Duck - initially negative towards storyboarding - admitted that it "turned out to be helpful" when their partner joined the session late. Similarly, Koala shared that the storyboards helped their group stay "on the same page" and they enjoyed collaborating on them. Three out of four teams commented in their interviews or surveys that they valued storyboarding as a communication tool.

Design Implications Since students find storyboards useful when planning for challenging features, a future planning tool may prompt students to consider how they can design their game to be

novel or challenging. Some groups added interesting features to their maze games - the simplest game - like timers and collectibles. Future templates may ask students to reflect on the uniqueness and enjoyment of their games. Because students enjoyed and valued storyboarding for its collaborative support, future planning tools should also support pair programming by allowing for concurrent editing similar to Google Slides.

7.3 Storyboarding struggles and solutions

Students expressed difficulty with knowing how to represent their ideas using storyboards. DZ explained that they struggled to "organize the amount of slides [they] were going to have" and often pondered "should I make these two separate scenes?" Koala shared that they were unsure how to represent a door opening and a ball bouncing. Instead, they relied on textual descriptions to express their ideas. The game mechanics that some students struggled to represent were easily constructed by others; e.g. DF represented a bouncing ball with arrows. Simple storyboard examples illustrating typical mechanics could help students create better scenes. Although Google Slides provides shapes that can help illustrate mechanics (e.g. arrows, callouts), adding them explicitly to a library of sprites and backgrounds may encourage students to use them.

The visuals in storyboarding may distract some students from planning complex mechanics. DZ spent a majority of their time on breakout creating a theme, leaving them without time to plan and program all the mechanics in a breakout style game. Providing students a sprite library organized by theme may prevent students from spending a majority of their time planning their aesthetics and give them time to plan more complex mechanics. For example, DF, who were satisfied with the graphics provided to them, used their time to create rich storyboards detailing their mechanics.

7.4 What planning supports do students need?

We expected students would first use storyboarding to generate ideas for visual and interactive game aspects, and then use PlanIT to plan their programs to achieve these ideas. However, all of the storyboarding groups opted out of using PlanIT for at least one of the games. The groups who used both storyboards and PlanIT used PlanIT as expected: to add game descriptions and define interactions between sprites. Although the storyboards did not ask students to plan coding structures, all groups used the scene descriptions to describe their code. DF and DZ wrote pseudo code, BK and WC described variables, and all groups defined conditionals.

BK, with low confidence, found storyboarding affirming as it facilitated their accomplishment of features they did not feel capable of implementing. The more confident DF and CW teams were initially negative about planning, but felt the storyboards were helpful as game complexity increased. The highly confident OR from Condition B found planning to be unhelpful, suggesting that in PlanIT they could not visualize their ideas. These findings suggest that the visual and collaborative support that storyboards provide can benefit novice programmers with varying levels of confidence.

The more confident students are in programming, the less benefit they perceive in planning. OR did not see value in planning and did not accomplish more than lower confidence groups who seriously engaged in planning. As suggested by prior work, more confident students may believe that block-based programming environments cannot be used for advanced programming [20]. These findings suggest that students, especially those with higher programming experience, need examples of effective storyboards and innovative games to realize the potential of these environments.

Table 3: Key design implications

Future storyboarding tools should…
allow students the flexibility to alter their designs quickly.
have visual and textual aspects of planning.
prompt students to adapt their game to be novel or challenging.
support pair programming by allowing for concurrent editing.
provide example storyboards that illustrate popular mechanics.
include a sprite library that is organized by theme.

8 CONCLUSION

Through a three-session exploratory study, our findings provide insights into how students perceive storyboarding for block-based games to support their creativity, the conditions in which students find storyboarding helpful, and challenges students experienced using both planning tools. This study provides recommendations for designing future planning tools for block-based environments.

We found that novice programmers think storyboarding aids in pair programming, supports their creativity, and allows them to easily alter their designs. Students find storyboarding helpful when the complexity of the games they plan for match or exceed their programming level. We also found that within our sample, novice programmers had a preference for storyboarding over PlanIT.

Although this research provides insights into students' experiences using storyboards, there are limitations. This study was conducted online and not in an authentic classroom setting; therefore, we are limited in our generalization of their experiences. Because the sessions were self-directed, students who storyboarded did not use PlanIT during every session. This limits our comparison of the affordances of the tools. Further studies may better illuminate how the tools can be used in combination for planning. Additionally, in our study design, there is no condition that only uses storyboarding which limits the opportunity to gain similar insights about storyboarding that we learned about PlanIT from Condition B.

Future work is required to understand how storyboarding affects student programming outcomes. Replicating this study in real classrooms may tell us how a wider population storyboards, as our sample consisted of highly motivated students with pre-established interests in computing. Finally, it may be useful to see how the use of storyboarding and PlanIT affects how students plan for future projects without the tools in a long-term study. The results may tell us what parts of planning students find useful and illuminate how to use these tools to promote the learning of planning skills.

ACKNOWLEDGMENTS

This work is based on NSF grant number 1917885. Any findings are the opinions of the authors and do not reflect the views and opinions of the National Science Foundation.

REFERENCES

[1] Carl Alphonce and Blake Martin. 2005. Green: A Customizable UML Class Diagram Plug-in for Eclipse. In *Companion to the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications* (San Diego, CA, USA) *(OOPSLA '05)*. Association for Computing Machinery, New York, NY, USA, 108–109. https://doi.org/10.1145/1094855.1094887

[2] Alexander Card, Wengran Wang, Chris Martens, and Thomas Price. 2021. Scaffolding Game Design: Towards Tool Support for Planning Open-Ended Projects in an Introductory Game Design Class. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, USA, 1–5.

[3] Veronica Catete, Amy Isvik, and Marnie Hill. 2022. A Framework for Socially-Relevant Service-Learning Internship Experiences for High School Students. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*. ACM, New York, NY, 815–821.

[4] Michael A Evans, Brett D Jones, and Sehmuz Akalin. 2017. Using Video Game Design to Motivate Students. *Afterschool Matters* 26 (2017), 18–26.

[5] Ilenia Fronza, Nabil El Ioini, and Luis Corral. 2017. Teaching computational thinking using agile software engineering methods: A framework for middle schools. *ACM Transactions on Computing Education (TOCE)* 17, 4 (2017), 1–28.

[6] Dan Garcia, Brian Harvey, and Tiffany Barnes. 2015. The beauty and joy of computing. *ACM Inroads* 6, 4 (2015), 71–79.

[7] Terrell Glenn, Ananya Ipsita, Caleb Carithers, Kylie Peppler, and Karthik Ramani. 2020. *StoryMakAR: Bringing Stories to Life With An Augmented Reality & Physical Prototyping Toolkit for Youth*. ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376790

[8] Joanna Goode, Gail Chapman, and Jane Margolis. 2012. Beyond curriculum: the exploring computer science program. *ACM Inroads* 3, 2 (2012), 47–53.

[9] Jamie Gorson and Eleanor O'Rourke. 2020. Why Do CS1 Students Think They're Bad at Programming? Investigating Self-Efficacy and Self-Assessments at Three Universities. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (Virtual Event, New Zealand) *(ICER '20)*. Association for Computing Machinery, New York, NY, USA, 170–181. https://doi.org/10.1145/3372782.3406273

[10] Brian Harvey, Daniel D Garcia, Tiffany Barnes, Nathaniel Titterton, Daniel Armendariz, Luke Segars, Eugene Lemon, Sean Morris, and Josh Paley. 2013. Snap!(build your own blocks). In *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM, New York, NY, 759–759.

[11] Wei Jin, Albert Corbett, Will Lloyd, Lewis Baumstark, and Christine Rolka. 2014. Evaluation of Guided-Planning and Assisted-Coding with Task Relevant Dynamic Hinting. In *Intelligent Tutoring Systems*, Stefan Trausan-Matu, Kristy Elizabeth Boyer, Martha Crosby, and Kitty Panourgia (Eds.). Springer International Publishing, Cham, 318–328.

[12] Hieke Keuning, Bastiaan Heeren, and Johan Jeuring. 2017. Code Quality Issues in Student Programs. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (Bologna, Italy) *(ITiCSE '17)*. Association for Computing Machinery, New York, NY, USA, 110–115. https://doi.org/10.1145/3059009.3059061

[13] Alexandra Milliken, Wengran Wang, Veronica Cateté, Sarah Martin, Neeloy Gomes, Yihuan Dong, Rachel Harred, Amy Isvik, Tiffany Barnes, Thomas Price, et al. 2021. PlanIT! A New Integrated Tool to Help Novices Design for Open-ended Projects. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. ACM, New York, NY, USA, 232–238.

[14] Kylie Peppler and Yasmin Kafai. 2007. What videogame making can teach us about literacy and learning: Alternative pathways into participatory culture. In *Situated Play: Proceedings of the Third International Conference of the Digital Games Research Association (DiGRA)*. University of California, UC Irvine, 8 pages.

[15] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.

[16] Peeratham Techapalokul and Eli Tilevich. 2017. Novice Programmers and Software Quality: Trends and Implications. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE T)*. IEEE, USA, 246–250. https://doi.org/10.1109/CSEET.2017.47

[17] Jakita O Thomas, Yolanda Rankin, Rachelle Minor, and Li Sun. 2017. Exploring the difficulties African-American middle school girls face enacting computational algorithmic thinking over three years while designing games for social change. *Computer Supported Cooperative Work (CSCW)* 26, 4 (2017), 389–421.

[18] Khai N Truong, Gillian R Hayes, and Gregory D Abowd. 2006. Storyboarding: an empirical determination of best practices and effective guidelines. In *Proceedings of the 6th conference on Designing Interactive systems*. ACM, New York, NY, 12–21.

[19] Ian Utting, Stephen Cooper, Michael Kölling, John Maloney, and Mitchel Resnick. 2010. Alice, Greenfoot, and Scratch – A Discussion. *ACM Trans. Comput. Educ.* 10, 4, Article 17 (nov 2010), 11 pages. https://doi.org/10.1145/1868358.1868364

[20] David Weintrop and Uri Wilensky. 2015. To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-Based Programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (Boston, Massachusetts) *(IDC '15)*. Association for Computing Machinery, New York, NY, USA, 199–208. https://doi.org/10.1145/2771839.2771860