Dance Tool: Creation of a New Culturally Situated Educational Game

Shaun Pickford
Software Information Systems
shaun.pickford@gmail.com

Dr. Tiffany Barnes
Computer Science
tiffany.barnes@gmail.com

University of North Carolina at Charlotte

Introduction

Culturally Situated Design Tools (CSDTs) are online tools that center on teaching a specific mathematical or computer science concept to grade school aged students (Eglash, Bennett, O'Donnell, Jennings, & Cintorino, 2006) via a cultural context. Through leveraging a cultural aspect that the target audience is likely familiar with, they have been shown to be an effective method for teaching basic Math and Computer Science concepts (Eglash, Bennett, O'Donnell, Jennings, & Cintorino, 2006). CSDTs sometimes fall short of effectively teaching advanced and broad concepts, and some researchers have explored the area of converting existing CSDTs into games to improve their effectiveness (Boyce & Barnes, 2010). Previous research shows that games foster learning by nature, which provided us the motivation to create a new and unique type of culturally situated game (Isbister, Flanagan, & Hash, 2010). Our approach combines the idea of CSDTs and games, and creates a new and unique Culturally Situated Educational Game that leverages dance and pop culture to teach grade school students foundational Computer Science concepts.

Problem and Motivation

The demand for Computer Science graduates is increasing steadily in today's marketplace, while universities are seeing declining enrollment in their computing programs (Zweben, May 2005). Our project, Game2Learn, focuses on targeting and retaining students in Computer Science (Barnes, Powell, Chaffin, Godwin, & Richter, 2007) through development of educational games. It is important that students have a positive attitude and interest towards the field of Computer Science, especially prior to entering college.

CSDTs, created by Ron Eglash at Rensselaer Polytechnic Institute, are successful in teaching basic math skills to grade school students (Eglash, et al, 2006) in a fun context. The subjects range from Cartesian Coordinates and graphing concepts seen in the Virtual Bead Loom to trigonometric functions used in BreakDancer (Eglash). The range of cultures explored in the CSDT project include: African, African American, Youth Subculture, Native American and Latino cultures (Eglash, Culturally Situated Design Tools). Of particular interest to us is the Youth Subculture.

Dance Tool draws on youth, pop and dance cultural aspects to create a sandbox-like environment where students can program their characters to dance. We believe that Dance Tool makes progress towards creating an educational game capable of teaching introductory computer science concepts while incorporating familiar cultural aspects to students at a critical development point in their academic lives.

Background and Related Work

An aforementioned CSDT, BreakDancer, enables players to program a virtual human to perform dance moves by use of trigonometric functions (Eglash, Culturally Situated Design Tools). Players specify a rotation or translation of individual body parts of their character for each frame of an animation. The complexity of trigonometry is apparent while playing BreakDancer, and the time required to make even a simple dance move in BreakDancer makes it difficult to create more complex dances. Another CSDT, Rhythm Wheels, allows players to create unique audio clips with musical instruments from popular musical genres such as hip-hop and rock (Eglash, et al, 2006).

Over the last few years, our Game2Learn program in partnership with the STARS Alliance (Dahlberg, Barnes, & Rorrer, 2007) has participated in many outreach opportunities with middle and high school aged students through multiple avenues such as apprenticeships, summer camps and tutor sessions. During our more extended outreach opportunities, such as apprenticeships where we visit students once a week for 10 weeks, we often have students play games our program has created, or experiment with CSDTs. We observed that BreakDancer and Rhythm Wheels were often the two most frequently chosen CSDTs by the students. This is likely due to the students relating more to the youth, pop and dance culture elements of these tools more so than the other tools. One of the major motivators behind CSDTs is that students need to be able to relate to the material (Eglash, Bennett, O'Donnell, Jennings, & Cintorino, 2006). We sought to combine the relatable concepts behind BreakDancer and Rhythm Wheels through use of similar youth, dance and pop culture aspects in Dance Tool while targeting the learning around fundamental computer science concepts.

Approach and Uniqueness

Dance Tool was developed in the Unity Development environment utilizing the JavaScript and C# programming languages and frameworks. (UNITY: Game Development Tool). The game places the player in the shoes of a choreographer creating a dance. There are two dancers the players can choose to control, Maya and Max (female and male, respectively). The process of writing out choreography for a dance is a creative and fluid process, much like the writing of a computer program. Prior research informs us that fostering the creative process is very important in creating a successful educational tool (Aragon, Poon, Monroy-Hernandez, & Aragon, 2009). To accomplish this in Dance Tool, we present the player with a blank Notebook on which they create their dance. The nature of Dance Tool presents the player with what is essentially a sandbox environment where they are only limited by their own creativity.

To add items to their Notebook, the player creates a Dance Instruction, which is a small grouping of Dance Moves. Dance Instructions can be named, looped and specified to start and end on a specific beat in the song. Instructions can be grouped together, and are then stored in Pages, which represent sections of the song such as Intro, Chorus, Bridge, etc. Pages are stored in the Notebook, which contains all information about the dance the player creates.

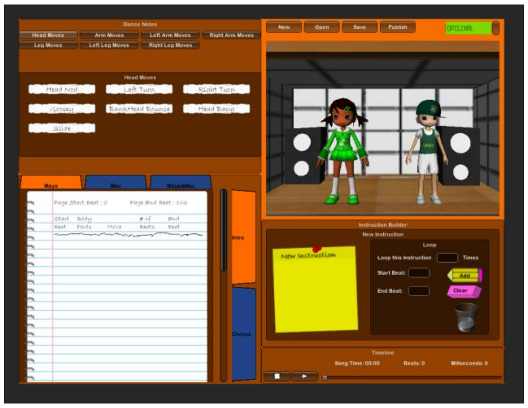

Figure 1: Dance Tool's Interface

The analogy of creating a completed dance by combining smaller components provides a very clear abstraction of the creation of an Object-Oriented computer program. Instructions are analogous to Classes and Objects in Computer Science, with the Dance Moves that make up the Instructions representing the code within a class. Pages represent an abstraction of Functions, and the Notebook represents a completed program. This abstraction provides the core learning goal of Dance Tool, which is learning the basic concept of a computer program.

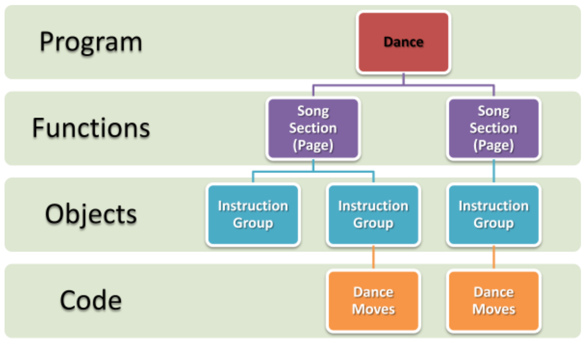

Figure 2: Abstraction in Dance Tool

At the start of play, students can choose from a selection of popular dance songs which have been pre-imported into the game. Each song is encapsulated in a unique data structure that stores relevant information about the song, including the audio file, beats per minute, song sections, and time markers. This information is vital to the program because it allows players to create sets of dance moves for specific song sections that can be reused. This is similar to most dance performances where the performer will repeat a series of dance moves during the chorus of a song.

Players drag and drop Dance Moves onto their Instructions, and simply click Add when they have completed their Instruction. The underlying data for all of this is saved in a multi-tiered data structure, with each level preserved in its own list for future editing.

We quickly realized that the multiple levels of data would cause a problem at run time. Each Dance Move contains a pointer to an animation for the character model to perform. Players can execute multiple animations at once as long as the body parts of the character model affected do not overlap (Dance Tool does this error checking as the player is creating their Instructions). Currently there are over 30 animations for each character model in the game. Each of these Dance Moves is encapsulated in an Instruction object. Finally all of these are stored in the highest level data structure, the Notebook. When the players choose to play their dance, the program has to iterate through all of these layers of data to find the animation that needs to be played for the current beat of the song. This iteration happens during each update loop, which can be multiple times per second. At this point, we had quite the time complexity issue on our hands.

With ‘n’ beats in a song, ‘m’ Instructions per beat, and ‘x’ Dance Moves per Instruction, our computational complexity for the underlying algorithm in Dance Tool was O(n*m*x). This was unacceptable, and was causing a large amount of lag when the players went to edit their Dances after creating them for the first time, as well as very shaky animation during run time.

To remedy this, a conversion method was implemented. The only pertinent data the program needs to know at runtime is the ID of the animation that needs to be played for the beat. A new object was formed called a Timestamp. A one-dimensional array called The Dance was created that held Timestamp objects. The length of this array was set to the number of beats in the song. The Timestamp object contains only the IDs of animations that should be playing for the specified beat, and a Boolean indicating if this beat is the start of a new animation. Once players finalize their dance, the multi-tiered data structure is converted into The Dance, which the game iterates through for each beat interval. The conversion does not change the structure of the Notebook, thus leaving the users dance intact for future editing. We also changed the functionality of editing to update the data structure for each change the player makes in real time, instead of when they finalize their dance, which significantly reduced the build time needed to create The Dance.

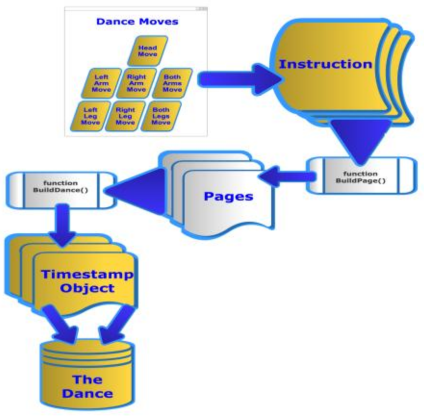

Figure 3: Visualization of data structure for Dance Tool

Results and Contributions

The programming of Dance Tool turned out to be a beneficial learning experience for the development team. While this is a positive effect, the purpose of Dance Tool is to teach basic Computer Science to its target audience of grade school students. Once the time complexity issue had been resolved, we sought to incorporate measurable learning elements into Dance Tool. Tutorials were created that guided the players through learning the Dance Tool interface, while also learning its abstraction of Computer Science in the process. For example, the first tutorial informs the students of the Notebook and its representation as the computational artifact they are creating. It describes the similarities to a computer program and the elements that a program comprises of. These details are elaborated further in the remaining tutorials.

Dance Tool also provides an environment to challenge players’ computational thinking skills. While they are learning about the construction of a computer program, an opportunity presents itself to guide the players into actually using these concepts within Dance Tool. We challenge the players to create a dance of a minimum length (such as 45 seconds) using the minimum number of actions possible. An action is logged every time the player adds something to the Notebook. With this style of challenge, we guide the players to think of how they can create a complex dance with minimal actions, thus guiding them to use Instructions and groups of Instructions. This is directly analogous to using Objects in a program to perform redundant tasks.

During the design process, one of our primary focuses was to provide minimal challenge from the interface of the game, as we want the players to focus on learning about the basics of a computer program and creating their dance (Isbister, Flanagan, & Hash, 2010). To measure the simplicity of the interface, Dance Tool was play tested with a summer camp of 21 high school juniors and seniors. The students played the game for 90 minutes and, after brief instruction on how to play, they were presented with a challenge to design the best dance. Most of the students participated and were able to effectively utilize the interface to create complex dances with minimal assistance. Afterwards, students were given a quick survey on their enjoyment level of the game, its simplicity, and suggestions for improvement.

Overall, how much did you enjoy Dance Tool?
"I really liked it"
"I liked it a lot!"
"It's Cool"
What Would You Change or Add to Dance Tool?
"Add way more moves and songs"
"I would put more dance moves and song selections"
"Song database and customizable outfits…volume control…"

Table 1: Sample Questions and Responses from Dance Tool Survey

Our work so far has provided us with a solid foundation for continued development. Currently Dance Tool is a sandbox style environment where players can create any dance they choose. The conversion of this environment into a fully playable and automated game is the next step in our development process. The tutorials provide an element of learning; however we want the learning to become more seamless within the game. We would also like to incorporate dynamically generated challenges that push the player towards creating computationally complex dances with minimal moves. Scoring of the dances also needs to be implemented, and our scoring system will need to reward players for making a complex dance with a minimal number of moves. In addition, we need to log all of this data and measure the learning activity during play.

Once the game components have been integrated, we next want to incorporate this into a social gaming website that our Game2Learn project has created called Community.Game2Learn. This site features collaborative games that can be played online, and provides an environment where players can share computational artifacts they have created, rate and comment on others content, and compete for global scores within the site. We believe the inclusion of Dance Tool on the Community.Game2Learn site, along with the automated scoring of the dances, will provide intrinsic motivation for the players to challenge themselves to create the best dances they can, thus improving their computational thinking skills.

Dance Tool has the potential to become a ubiquitous learning tool that grade school teachers can use in their classrooms to teach fundamental Computer Science concepts. It provides elements of creativity, fun, collaboration and competition in a familiar cultural context to the students. In combination with other educational games on the Community.Game2Learn site, Dance Tool can become an important aspect in teaching basic computing concepts as well as fostering positive attitudes towards the field of computing at a critical time in a student's academic development.

Future Work

Currently we are working on the addition of game elements such as scoring, player challenges, and multiple modes of play. We also plan to perform a qualitative feedback study once these changes are implemented. Next, Dance Tool will be integrated into a Community.Game2Learn website where players will be able to publish and share their custom dances with other players and receive feedback and collaboration.

After integration of the game and collaborative elements, we plan to run user studies to measure the learning gains of the concepts taught in Dance Tool. Our study design will consist of three groups of at least 30 middle school aged students in each group to ensure statistical significance. They will be presented with a pre-test to measure their base knowledge of the components of a computer program and object-oriented design. After this test, one group will be given Dance Tool to play and will be challenged to create the highest scoring dance that they can and publish it to the Community.Game2Learn site. The second group will be given the sandbox version of Dance Tool to play with, but with no challenges presented. They will also be allowed to play others dances from the Community.Game2Learn site. The third group will be given a brief lecture on the same concepts that Dance Tool teaches. Afterwards, we will give each group a post test with isomorphic questions on the same concepts from the pretest. It is expected that the group that played Dance Tool with the challenges incorporated will have the highest learning gains from pre to post test. We hope to prove that Dance Tool, along with the computational thinking challenges incorporated into the game, will show stronger learning gains than a traditional lecture or an unguided play session that students currently experience when using CSDTs. This information will guide our future work and provide us with the knowledge needed to distribute Dance Tool to K-12 teachers for use in the classroom. We hope for Dance Tool to become an effective teaching tool that will assist in teaching the foundational concept needed for success in the Computer Science field.

References

Aragon, C., Poon, S., Monroy-Hernandez, A., & Aragon, D. (2009). A tale of two online communities: Fostering collaboration and creativity in scientists and children. *ACM C&C*, (pp. 9-18). New York, NY.

Barnes, T., Powell, E., Chaffin, A., Godwin, A., & Richter, H. (2007). Game2Learn: Building CS1 Learning Games for Retention. *ITiCSE* (pp. 121-125). New York, NY: ACM.

Boyce, A., & Barnes, T. (2010). BeadLoom Game: Using Game Elements to Increase Motivation and Learning. *FDG '10: Proceedings of the Fifth International Conference on the Foundations of Digital Games.*

Dahlberg, T., Barnes, T., & Rorrer, A. (2007). The STARS leadership model for broadening participation in computing. *Frontiers in Education Conference.*

Eglash, R. (n.d.). Retrieved from Culturally Situated Design Tools: Teaching Math Through Culture: http://www.rpi.edu/~eglash/csdt.html

Eglash, R. (n.d.). *Culturally Situated Design Tools*. Retrieved March 2011, from http://csdt.rpi.edu/

Eglash, R., Bennett, A., O'Donnell, C., Jennings, S., & Cintorino, M. (2006, June). Culturally Situated Design Tools: Ethnocomputing From Field Site to Classroom. *American Anthropologist* , 347-362.

Isbister, K., Flanagan, M., & Hash, C. (2010). Designing games for learning: Insights from conversations with designers. *ACM CHI*, (pp. 2041-2044). New York, New York.

UNITY: Game Development Tool. (n.d.). Retrieved from http://unity3d.com/

Zweben, S. (May 2005). *Computing Research Association Taulbee Survey.*