UbiPlay: An Interactive Playground and Visual Programming Tools for Children

Jussi Mattila and Antti Väätänen
VTT Technical Research Centre of Finland
Sinitaival 6, Tampere
P.O. Box 1300, FI-33101 VTT, Finland
+358 40 592 7979
jussi.mattila@vtt.fi, antti.vaatanen@vtt.fi

ABSTRACT
Children develop important skills at playgrounds. Physical play promotes health and reacting to other children establishes social behavior patterns. By augmenting playground elements with sensor technology, video displays, and computer software, we pursued to take the experience further. This paper describes what was achieved; UbiPlay, a technology platform for programmable interactive playgrounds. UbiPlay allows children to create and play games in interactive playground environments. We present a play space built using the technology and results from qualitative evaluations, performed with 44 school children between the ages of 10 and 12. Results indicate that end-user programmable playgrounds like ours can provide much stimulus and excitement for children.

Keywords
Interactive environments, visual programming environments, playgrounds, children

ACM Classification Keywords
H.5.2 [Information Interfaces and Presentation]: User Interfaces – Graphical user interfaces (GUI), Interaction styles; D.2.6 [Software Engineering]: Programming Environments – Interactive environments.

INTRODUCTION
Outdoor play and well-designed playgrounds have been shown to promote children’s physical, social, and mental development [13,22,26]. Though most playgrounds have so far remained free of wires and transistors, technology has invaded other areas of children’s lives. From high-tech musical toys [16] to programmable LEGO Mindstorm kits [18] to mobile phones, children are interacting with pervasive sensors and computers from an early age. Unfortunately, not all technology is considered good for the children. It has been speculated that they spend less time outdoors participating in physical activities because of the exciting interactive worlds inside the devices [6]. Some studies are even linking technological advancements of recent decades to the poor physical state of children [2].

Another issue with interactive entertainment and children is its effect on their cognitive and social development. Though researchers struggle to answer the broader questions, it has been shown that interactive media can have a positive effect on learning in narrow contexts [15]. Likewise, the social influences of interactive entertainment are not seen as all negative. Collaboration and sharing of experiences in the virtual worlds can enrich children’s social lives [24].

We believe that it is possible to create interactive entertainment that fosters the beneficial aspects and reduces unwanted consequences. In particular, the generally positive influence of playground activity could be united with interactive entertainment to create novel interactive playgrounds. If interactive playgrounds were available, some children might be lured back outdoors to play physical and social games. If the children also had control over the content in the interactive playground, it could further add to the playground’s appeal, enhancing its social aspects and fostering communities of similarly minded children. End-user customizability might also draw some children into programming activity because the rewards of programming would be more relevant to them. In this way, the system would also function as a learning tool for children. The goal for UbiPlay was to develop a playground technology platform with visual programming tools for children that would pave the way for exciting interactive playgrounds. Our vision is that a well-working blend of the two existing concepts, physical and social play in the playgrounds and interactive programmable environments, could in many ways prove better for the development of children than contemporary flavors of interactive entertainment.

RELATED WORK
To our knowledge, technology platforms for end-user programmable physical playgrounds have not been researched or developed previously. Closely related research on programmable play spaces [20] and mixed reality gaming environments [19] have validated that the
combination of physical play equipment and computer software can be intriguing to children. Relevant research avenues in virtual and physical interactive environments, programmable toys, and programming languages for children have also been considerably fertile for a long time.

Interactive environments
Despite residing only inside computers, virtual reality playgrounds have common interests with what we want to achieve. Some have had a more modest goal of just allowing children to play together in a virtual environment [17], but most are focusing on the educational aspects of collaborative work and play [10, 25]. Interactive physical environments preprogrammed by adults have also been researched to some extent [3, 14]. As a result of this research, interactive worlds that are not confined to the desktop can already be found in everyday life settings such as in museums and amusement parks. However, programmable play spaces have not yet materialized in the play areas of malls, restaurants, and stores, or at the playgrounds. The lack of customizable interactive environments for children is somewhat surprising, since providing children with control over stories has long been an incentive for researchers to design and develop physical interactive storytelling environments [1, 8, 27]. To change this trend, we wanted to create a customizable platform to build end-user programmable interactive playgrounds with.

Programming children’s interactive environments
Several programming environments have been designed to make programming more accessible to novice programmers of all ages [12]. Some of them have empowered young users of interactive environments by utilizing a programming paradigm called physical programming, which is the act of arranging physical items to achieve programming tasks [20]. Outside of physical programming, programming environments for children usually rely on visual programming, which offers a higher abstraction level than programming languages based on text syntax do. Visual programming environments provide end-users with control over the interactive spaces by employing flavors of data flow systems, rule-based systems, and finite state machines (FSM), or a combination of these. Data flow systems are programmed by connecting icons or manipulating objects to achieve desired reactions, see e.g. [5, 11]. In rule-based systems, the programmer specifies rules by demonstrating a postcondition on a precondition. Logic emerges when rules are repeatedly followed during the execution of a program [23]. FSMs are defined as a model of dynamics, containing states and transitional links, where the model (i.e., machine) stays in a state until a condition for a transition becomes true. At such a time, the transition causes the state machine to change its state [7].

IMPLEMENTATION OF THE UBIPLAY PLATFORM
In our view, to build a platform that unites a playground with an end-user programmable interactive environment, four components are needed:

- The playground.
- Sensors and actuators embedded in the playground.
- Software for interacting with the sensors and actuators.
- Programming tools that give control over the sensors and actuators.

The first three components constitute a minimum set of hardware and software that will allow a playground to react intelligently to users’ actions. The last component allows end-users to modify the required actions and the playground’s reactions. Our technology platform for interactive playgrounds, coined UbiPlay, comprises of the last three points in the list above. Before starting implementation work, we assembled hardware for the initial UbiPlay playground in our research laboratory using off-the-shelf products (see Figure 1). Pressure sensitive floor tiles were used for monitoring the play space. As actuators, a video screen and a mobile phone were used for displaying graphics and text, and a loudspeaker was used for sounds. Using this hardware as a reference, we began developing the technology platform. Implementation proceeded in two phases. The first phase consisted of creating software components that interact with sensors and output devices embedded in playgrounds, and creating some simplistic games to verify that these components work as intended. In the second phase we developed a proprietary visual programming language for children. When used in conjunction with the hardware interaction components of the platform, the visual programming tools allow children to create stories and games for playgrounds built with the platform.

Hardware
The reference playground in our facilities consists of 36 pressure sensitive floor tiles organized in a square shape measuring 3x3 meters, a video screen, and loudspeakers. The heart of the interactive runtime platform is a Windows XP computer, the game server, which runs the UbiPlay platform software and consequently controls all the components.

The tiles are solid rubber playground tiles. To embed sensors in them, each tile’s bottom side was carved hollow and a pressure sensor was inserted into the empty space. A

Figure 1. The interactive play space in our facilities.
wire runs from each sensor into an I-CubeX Digitizer module (by Infusion Systems Ltd., Montreal, Canada), which converts analog voltages generated by the sensors to MIDI messages. We use two digitizers to avoid the input limit of 32 sensors in one I-CubeX Digitizer. Both digitizers are connected with MIDI cables to a USB MIDI hub attached to the game server computer. In addition to the pressure sensitive floor tiles that we used, the digitizers allow for numerous other types of sensors including light, proximity, orientation, and wind sensors. The main visual output device is a large video screen, but we also have mobile phones for displaying applications more privately. Audio is handled directly by the game server computer, which is connected to an active loudspeaker placed near the playground (see Figure 2).

Hardware is needed for all interactive playgrounds, but it can vary greatly from one playground to the next. That is why the UbiPlay platform can, and should be customized to use whatever hardware a playground is built with. The platform provides a hardware abstraction layer which allows it to interact with any devices in the playground, as long as a software module describing that particular piece of equipment has been implemented.

Interaction software
Software development for the platform began with a playground sensor configuration utility. It allows system installers define what types of sensors exist in the playground and also their physical locations. To make use of any sensor signals in games, a modular signal processing software component was created. When people act on a playground, signals from the playground sensors are fed into the system. Plug-in modules interpret the signals in real-time, and when a plug-in notices a feature it is looking for, it creates an event that propagates out of the signal processing system to other parts of the platform for utilization. In our research laboratory, where the initial

![Image](50x583 to 288x750)

**Figure 2. The reference UbiPlay hardware.**

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tile entered</td>
<td>Occurs when a person steps on a tile.</td>
</tr>
<tr>
<td>Tile exited</td>
<td>Occurs when a person steps off a tile.</td>
</tr>
<tr>
<td>Jump on tile</td>
<td>Occurs when a person jumps on a tile.</td>
</tr>
</tbody>
</table>

**Table 1. Implemented sensor actions.**

UbiPlay implementation was built, we chose to expose three events for controlling the games (see Table 1).

Modularity was also extended to visual feedback. The platform does not explicitly display graphics on a monitor attached to the game server computer. Instead, it utilizes a programming interface to send text and images to any display clients attached to it. All display devices can be controlled using the same programming interface. As a consequence, they are interchangeable and it is possible to use more than one display at the same time. In our setup, a display client program exists as a separate process in the game server computer. To control the client, we have an implementation of the display interface over TCP/IP (Transmission Control Protocol/Internet Protocol) that was originally designed to transmit text and images over to another computer. In addition, we developed an implementation of the display interface for Bluetooth wireless connection technology to communicate text and image data to compatible mobile devices. The client for Bluetooth communications displays was implemented for the Symbian Series 60 mobile phone platform.

Creating games in the UbiPlay platform
Our view of creating technology is consistent with the notion made in [9], where it is suggested that most technology developed for children could make use of their imagination and ideas during the design phase. Thus, the most essential component of the UbiPlay platform is the Game Creator software. Its purpose is to expose input and output functionality of the interactive playgrounds for children to customize. Game Creator’s visual programming environment (see Figure 3) lets children make stories and

![Image](311x133 to 544x289)

**Figure 3. Visual programming environment. Eight state nodes have been connected with transitions.**

games that react to events occurring in the physical space and give feedback using, for example, text and audiovisual cues.

The process of creating games is simple:
1. Launch the Game Creator application on a Windows computer.
2. Create a game in the Game Creator’s visual programming environment.
3. Test the game within the Game Creator application.
4. Save the finished game to a file.
5. Upload the game file to the game server running the UbiPlay platform.
6. Play the game in the interactive playground.

Visual programming environment for children
Looking at previous research concerning children and programming, it was clear that simplicity is the key for getting successful results. Unfortunately, we had to dismiss physical programming, because the risk of losing physical programming props from a public area, which the platform targets, was considered too high. We concluded that a visual programming language would be expressive enough for our purposes and hopefully simple enough for children to use. Ultimately, we chose to create a visual programming environment utilizing finite state machines (FSM) due to them being easy to implement and easy to extend with new states. We also believed that programming the physical environment with FSMs would be meaningful to children, since presenting sequential challenges for players in the playground should be relatively easy concept to grasp.

Like most tools created during the development of UbiPlay, Game Creator’s visual programming environment is modular. It is possible to add and remove states available to programmers at will, and by doing that, allow or disallow functionality. After some consideration, seven states were made available for the programmers of the initial UbiPlay system (see Table 2). Unfortunately, limited space in the laboratory prevented us from having states that require running or climbing. Another issue that made us reduce the number of states was our goal to keep the game programming phase as simple as possible.

Creating a game using the Game Creator is done by dragging state nodes from a palette on the left side into a design view on the right side. The states are then programmed and connected to each other with directed links; transitions (see Figure 3). The transitions describe a game’s logic by forming branching paths that advance from a start node to an end node. To program a state, the user clicks on a configurable item and changes its value. All features of the programming tool can be accessed with a mouse except for entering text, which requires typing. Keyboard shortcuts also exist, but the user interface is designed to be mainly mouse driven.

### Table 2. The programmable states.

<table>
<thead>
<tr>
<th>State name</th>
<th>Purpose and programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display text</td>
<td>Displays text on a display device. Programmer types in text.</td>
</tr>
<tr>
<td>Display picture</td>
<td>Displays a picture on a display device. Programmer selects a picture.</td>
</tr>
<tr>
<td>Play sound</td>
<td>Plays a sound at the playground. Programmer selects a sound.</td>
</tr>
<tr>
<td>Step on tile</td>
<td>Waits until a person steps on the correct tile. Programmer chooses the tile on which to step on.</td>
</tr>
<tr>
<td>Jump on tile</td>
<td>Waits until a person has jumped on a tile for n times. Programmer chooses jump count and the tile to jump on.</td>
</tr>
<tr>
<td>Find path</td>
<td>Waits until a path of tiles ([1...n]) has been walked on. Programmer chooses the tiles that form the path.</td>
</tr>
<tr>
<td>Stand on tiles</td>
<td>Waits until a person or persons are standing on n tiles. Programmer chooses the tiles.</td>
</tr>
</tbody>
</table>

The majority of information about a state can be seen whenever the state is visible in the design view. Only text fields describing the plot of the story are truncated to fit into the small space reserved for them. Depending on the state, programming it may require the user to enter text or select sensors, pictures, and sounds. When selecting sensors, a dialog with a visual representation of the sensors on the playground is displayed. Likewise, the pictures for visual feedback and the sounds for audio feedback are selected from separate dialogs (see Figure 4). Information about the locations and types of sensors in the playground is retrieved from the UbiPlay platform and bound to the Game Creator during load-time. Thus, invalid values cannot be entered and setting up the programming environment is not

![Figure 4. Sensor, picture, and sound selection dialogs.](image)
something children need to worry about when creating games.

The most important visual cue in the programming environment is coloring. The colors are fully configurable, but for evaluations we chose to use red for anything that is still incomplete and green for valid items. Another helpful feature in the Game Creator is the cursor that changes shape whenever it is on an item that can be selected or edited. To allow testing of the game logic during programming phase, a component running the game locally is integrated within the editor. If there are any incomplete items when starting a game, a verbose message is given and the user is instructed to fix them. Games created with the Game Creator can be stored to disk as a package file that describes the logic of the game, the locations of the states in the design view, and any resources being used. This allows users to modify a game on any computer with the Game Creator software and finally, allows uploading the game to the game server computer for playing it in the interactive playground.

Evaluation
After the implementation work and the first trials within the project team were finished, we set out to evaluate the UbiPlay platform with school children. The testing period extended from February to March in 2005. The primary goals for the evaluations were to find out possible usability problems in the hardware and software, define how the Game Creator is used for making games, and how children feel about playing games they have created using the UbiPlay platform. As a subtext to the evaluations, we presented the interactive playground as a possible future teaching aid in schools.

Users & Methods
Overall 44 children varying in age between 10 and 12 years and representing evenly both sexes participated in the evaluations coordinated by three researchers. To get as much information as possible in the time we had, we chose to test the UbiPlay platform with older children who are fluent readers and writers. Based on previous research concerning programming languages for children and interactive environments [8,20], we believe that the programmable parts of the UbiPlay platform could be suitable for younger children as long as they are literate. The interactive playground, on the other hand, can attract children of all ages. The participants were pupils of a comprehensive school in Tampere, Finland. They filled a background questionnaire which contained topics ranging from the use of computers to video gaming habits. Almost all of the children had a computer at home and all of the children had used computers in school. When asked about playing habits of computer and video games, boys were more active than girls. Boys’ favourite genres were sports games and driving simulation games. Among the girls ‘The Sims’, a life simulation game, was the most popular. Both genders used the Internet for information retrieval, and girls frequented chat rooms more than the boys. These findings were in line with much larger studies discussing children and interactive media [24]. In addition to testing the UbiPlay system with the children, four teachers from the same school were interviewed to hear their thoughts on the system and its value in education. All of the participants visited the facilities twice, and both times they spent approximately two hours using the UbiPlay platform.

During each evaluation afternoon, half of one class, up to 15 children, came to our facilities. They were then further divided into smaller groups of three to five children (see Figure 5). Each group operated a personal computer running the Game Creator software and had access to a researcher if they wanted to ask questions. The idea behind making games in small groups was that each group designed games to the other groups. Even though the programming environment was not designed to be a multi-user environment we hoped that programming in groups would encourage teamwork and allow all users to be active in the game creation process. All evaluation afternoons with children consisted of several game design sessions, each of which lasted for approximately thirty minutes. Groups were expected to design and program one game during each thirty minute session. The design sessions were kept short intentionally; the ultimate goal of the UbiPlay platform is to get children play and socialize at the playgrounds, not sit in front of a computer. After a design session, all groups gathered in the laboratory with their finished games and members of other groups played the games in the interactive playground.

The game design sessions became gradually more difficult. In the first session, the concept of a programmable interactive playground was introduced and all groups developed a simple game adhering to instructions given beforehand. No instructions were given on how to use the visual programming tools. Instead, the children were given a development task like ‘Create a game in which the player is instructed to step on a certain tile, and when the player steps on the tile he or she is congratulated with text, pictures and sounds’. After having played the games created during the first session, a second game design and programming session took place. This session was spent creating an educational game that would teach players
mathematics, biology, or music. Subsequent to playing the educational games, in the third and final session of the evaluation afternoon, the groups were encouraged to use their imagination and create their own stories and interactions for the playground.

After a fortnight, the children came back for another afternoon of designing and playing games with the UbiPlay system in our facilities. This time they were allowed to spend all sessions exploring their own ideas, using the Game Creator to create unique games for each other. Finally, after the second afternoon of evaluations for a particular group of children had ended, they were engaged in group interviews [4,21]. Conversation and questions were focused on the following topics:

• Creating stories and games with the Game Creator.
• User interface and interaction with the playground.
• Interactive playground as part of a school day.

Results

The idea of making games and playing in the play space was learned quickly and crucial usability problems did not occur. Despite the short amount of time reserved for each individual game design session, the children were always able to create working games. Even so, we observed that more help for children would be appropriate in the visual programming environment. Especially in the beginning, a pedagogical agent described in [10] could aid in getting children familiarized with the programming environment. Since we could not implement the agent after the evaluations had began, we substituted it by manually showing how to drag a state from the state palette into the design view and how to connect it to another node. After the researcher had done this, the children took over. In a few minutes, they were well versed in using the Game Creator and could create adventures with little or no supervision. In the most ambitious games, the children began having trouble in arranging the states in the design view (see Figure 6). Like in [20], we noticed children taking different roles during the game design and programming sessions. Some were also more interested in creating games than actually playing them, and vice versa.

The true challenge for children in programming was developing logic and structure to the interactive games. Instructions and feedback in the games they created was not always adequate enough or it was easily misunderstood. Some of the games were virtually impossible to complete because the logic of the game was faulty; either no explanation was given for a task or a task was explained poorly. This usually led to a player getting stuck in the game, and eventually not being able to finish the game. There were also some groups who created games that insulted the players when they failed to complete a task in the game. This is an issue that should be addressed in the UbiPlay platform.

In general, children created games with goals related to mystery solving, and rescue or escape missions. Most of the games were short but some were of considerable length (see Figure 6), taking several minutes to play through. Branching possibilities in the programming environment were rarely utilized, which meant that the storylines were mostly linear. We limited the amount of pictures and sounds in the resource dialogs to only 21 bitmap pictures and 18 sound clips to choose from. Depicting mainly animals and musical instruments, the lack of options affected some of the stories. Nevertheless, they ended up being more imaginative than we had anticipated, featuring ghosts, zombies, classmates, and pop stars. In addition to jumping on a tile, one of the most common tasks employed by the children was to walk a path on the outermost tiles around the play space. They also created games requiring collaboration of several players. For example, a task programmed by one group instructed four people to stand in the corners of the play area before the game would proceed. In the end, the children wanted to have a lot more choice of pictures and sounds. They were hoping to add media to their games from the Internet. Some groups were also interested in making their own graphics and sounds with the platform. This was a feature that gained support from the teachers, who thought that custom content creation tools would combine even more traditional learning methods, like drawing, to the novel interactive environment.

Technology-wise, the UbiPlay platform worked well. Only the sensitivity of the floor tile sensors required some modifications. The floor sensor equipment that we use expects users to put a foot down in the center region of a floor tile, or the step will not register in the platform. During the evaluations, this was sometimes a source of frustration for the children. After the first evaluation afternoon, the sensors were adjusted to be more reliable for the light-weighted children. A sound cue was also added to indicate when a user successfully stepped or jumped on a tile. To remedy the problem completely, different playground hardware should be used. It is a viable option,
because the UbiPlay platform was designed to support even radical changes like this.

Both children and teachers considered the UbiPlay system a fun, novel, and motivating way to play and learn. The children remained very interested for up to two hours during the evaluations, making games and playing them in the play space. When asked what was the most fun, both programming and playing games received approximately equal amounts of support. Programming in small groups was perceived to develop teamwork skills and designing stories called for creativity and imagination. Playing the games created by others while they were watching was a social event. Most of the games designed by the children required some mental and even physical effort, which bodes well for larger playgrounds where there is space to run and climb.

The teachers, who evaluated the UbiPlay system as a future teaching aid, reckoned that making games with the Game Creator software was a part of a learning process. The subjects that teachers considered most appropriate for the UbiPlay system were mathematics, language studies and physical education. Teachers also emphasized that the system should be easy to move and assemble and should primarily be used in a classroom. Children suggested that besides classrooms, a system like this could be installed in a gymnasium or a lobby area.

CONCLUSIONS
We elevated the playground experience to new levels with the UbiPlay platform and were able to reach many of the goals set for it. The platform was found easy to use and extend and most importantly a lot of fun for the children. Children participating in the evaluations seemed enthusiastic about the possibilities and the qualitative interview process strengthened this view. The children were working in groups, played physically and socially engaging games, and at the same time they were being entertained by their own creations in a new interactive environment. Teachers were of the opinion that they could use a system like this as a teaching aid in programming and in theoretical subjects. Unfortunately, a perfect balance between programming activity and physical and social play in the playground was not yet achieved. In its current form, the system is perhaps more of an aid to learning programming skills than an enabler of rigorous physical play, which is our ultimate goal.

Having a configurable, extendable, and modular platform to build interactive playgrounds with can considerably shorten the effort required in creating them and making them available for the public. When interactive playgrounds become reality, we believe that they can have a positive effect on children’s development. At that point their potential in enhancing children’s lives should be examined in much greater detail.

FUTURE WORK
Despite positive feedback, there were a number of things that would improve the UbiPlay platform. Children saw both programming and playing as equally engaging activities, thus programming should be blended more intimately with playing in the physical environment, where we would want most activity to occur. To facilitate this, the programming environment should be revised heavily or even moved away from the desktop and into the playground environment. While that thought is very intriguing, we are currently focusing in two less laborious paths to follow. First, we are aspiring to solve the few issues in the hardware and software that were found during the evaluations. Second, we want to extend the currently implemented system with new features.

The process of making games must be effective. Wizards for setting up simple games would provide a better starting point than an empty design view. Graphing algorithms and clustering methods could help remove clutter in the design view, as would the ability to group multiple states together for more complex behavior. Also, better feedback and information about logical errors during programming and testing phases could prevent problems in game logic that were only encountered later on in the interactive playground. As in all environments where the users create content, the question of content moderation came up and should be addressed in the Game Creator application or the game server.

Development of new features should look into adding new hardware options, new types of states to the programming environment, and personalizing the experience. New programming states could provide programmers with more branching possibilities, timers, rhythmic challenges, speed challenges, and new cooperative tasks. Regarding personalization, we have plans to look into user identification methods and supporting custom pictures and sound effects. The latter, in addition to being a frequently requested feature, will provide better means for generating custom educational content. We also plan to have a programmable interactive playground with swings, slides, climbing frames, and sensors installed in actual everyday use. When that occurs, the playground should be subjected to exhaustive studies on how it will best benefit the children.

ACKNOWLEDGMENTS
This work has been funded by TEKES - Finnish Funding Agency for Technology and Innovation, Lappset Group Oy, VTT - Technical Research Centre of Finland, and TeliaSonera Finland Oyj.

REFERENCES


