SIGCHI Extended Abstracts Sample   
File: Note Initial Caps

[Paste the appropriate copyright/license statement here. ACM now supports three different publication options:

• ACM copyright: ACM holds the copyright on the work. This is the historical approach.

• License: The author(s) retain copyright, but ACM receives an exclusive publication license.

• Open Access: The author(s) wish to pay for the work to be open access. The additional fee must be paid to ACM.

This text field is large enough to hold the appropriate release statement assuming it is single-spaced in Verdana 7 point font. Please do not change the size of this text box.

Each submission will be assigned a unique DOI string to be included here]. (<= which license do we put here?)

# Abstract

First Author

TU Delft

Delft , NL

[D.Dumeljic-1@student.tudelft.nl](mailto:D.Dumeljic-1@student.tudelft.nl)

Second Author

TU Delft

Delft , NL

[D.J.P.vanPeer@student.tudelft.nl](mailto:D.J.P.vanPeer@student.tudelft.nl)

Third Author

TU Delft

Delft , NL

J.M.Raes@student.tudelft.nl

Fourth Author

TU Delft

Delft , NL

[O.D.F.Dikken@student.tudelft.nl](mailto:O.D.F.Dikken@student.tudelft.nl)

Fifth Author

TU Delft

Delft , NL

S.N.Dumasy@student.tudelft.nl

Queues have been around since the law of the jungle has been abandoned, and they are outdated, boring and time consuming structures. Time spent waiting is time wasted and possibly even frustrating if the waiting process does not occur in comfortable conditions. Entertaining people present in such a situation would improve their quality of life and could make them have a more positive mindset towards the event they were queuing for. Our solution not only entertains queue participants but also allows them to experience a group dynamic with other people in the queue making them enjoy the company of their fellow queue members instead of wishing they were the only one there. Who would have thought queues could ever become enjoyable experiences instead of time consuming structures.

# Author Keywords

Collaborative gaming; social gaming; motion-controlled gaming; Camera detection; Mini game; Physics based gameplay;

# Game Concept

The main objective of Penguin Peril is to acquire as many fishes as possible. But beware, evil is lurking and needs to be avoided at all cost. This can be achieved by altering the height of the wave the penguins slide over. When a penguin does not manage to avoid an obstacle, the count of the acquired fishes is reset to 0. The difficulty of the game also depends on the number of acquired fishes. The difficulty of the game mostly entails what kind of obstacles spawn and how fast they do. There are also several power ups that penguins can collect which alter the game.

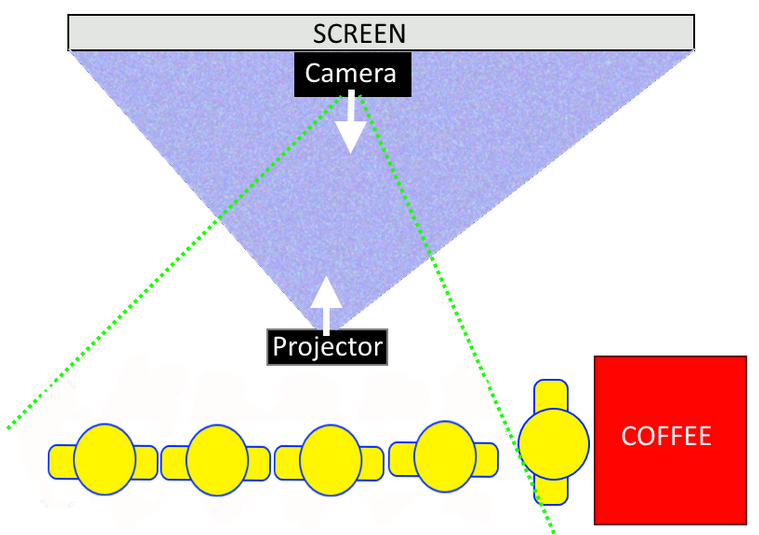
***characteristics***

Figure : Top view game setup.

The game is to be played by a section of a queue. It should motivate queue participants to work together as a team to achieve progression. The use of external devices such as audio and camera description or motion sensing is recommended.

***constraints***

The game must work in practice. A real life queue will be able to accept the setup and the game will function as intended when being played by a group of inexperienced queue participants that are actually waiting in line for their turn. Therefore the game must respect queue dynamics, being that players will be able to drop in and out at any given moment without having negative impact on the gameplay or influencing the pleasure of other participants. The software must have proper architecture, meaning that it has clean robust structure, all functionality is tested and the code follows many known design patterns. Also it must be written in java.

***challenges (design and technical)***

When you think of playing a computer game with a group of people you don’t often imagine it being in a public place, let alone whilst waiting in line. Not every game is appropriate to be played in such conditions. Since our game uses motion detection controls we also considered that it might not be inviting for a section of a queue to start waving around to make things move on a screen. To make the game work in a queue it has to be easy to drop in and out of meaning that we had to scrap character progression, level progression, item collection and story based games due to the fact that the game always has to be in the same state for people to experience it the same way. Finally queues come in many forms and lengths and making a game controlled by camera detection that works on any universal queue was in our case virtually impossible. This is why we limited ourselves to a straight segment of a queue, having a static background as to allow the detection algorithms to only detect the players (and not people/movement in the background).

***main idea (game design, game mechanics)***

Project Wave is a game in which a segment of a queue controls a curve using motions. Objects flow in one direction along the curve and need to reach targets avoiding obstacles along the way.

Our gameplay differs from that of a conventional game in that each player controls different parts of the same object. The players’ positions are recorded by a camera, of which the information is used to determine the current height of the players. The camera translates points defined by the player motions into game coordinates which are used to draw a curve. This curve represents a wave along which all objects roll. This wave is the only gameplay element directly controlled by the players.

The screen displaying the game lies parallel to the queue. The game direction, i.e. the direction in which the objects are rolling, is determined by the direction the queue is moving in.

The main gameplay elements are objects that continuously spawn, rolling along the wave in the game direction. They are introduced at the start of the screen and are destroyed when exiting the screen. The players dynamically control the shape of the wave which changes how the object moves along the wave: slopes affect the speed of the object, yet the object cannot be brought to a halt, nor can it roll backwards. Thus, players can form hills and valleys to influence the objects’ trajectories.

Other gameplay elements include targets and obstacles. Targets enter the right side of the screen and slowly progress towards the left side of the screen, as do obstacles. The players are meant to shape the slope in such a way that the objects hit the targets and avoid the obstacles.

Another gameplay element is the combo count. When an object hits a target a combo count is either initiated or incremented. Maintaining a combo is the main gameplay goal of the players. A combo is broken (reset to 0) when an object collides with an obstacle or an object misses a target (i.e. if the gameplay direction is from left to right, once the object’s horizontal position is more to the right than a target’s horizontal position we conclude the object has missed the target). As the combo count increases, the game becomes more difficult. Various gameplay elements will change, like the spawning of power-ups, object spawn time, object general movement speed, amount of targets/obstacles and positioning of spawn areas of targets or obstacles.

Power-ups are what makes reaching higher combos much more exciting.They will  appear depending on the combo count and can be hit to activate. As the last gameplay element, power-ups open up a myriad of possibilities. Inverting controls, inverting game movement, various modes like frenzy, with lots of objects spawning at once, or wacky, where the world colors turn psychedelic and obstacles become targets and vice versa. etc, etc.

When the combo count is reset, the game does not enter a ‘game over’ state, the only loss the players can encounter is the end of a combo. Subsequently, the difficulty is also lowered, and activate power-ups will become inactive.

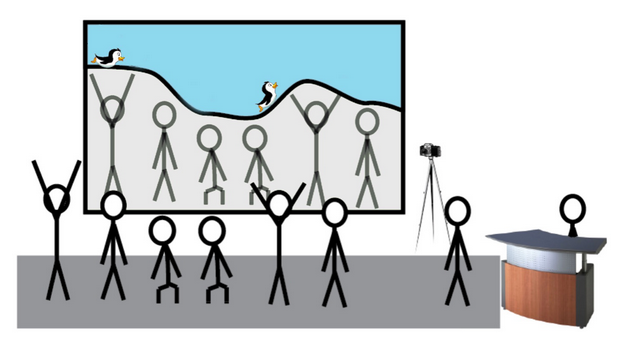
With respect to queue dynamics, the game does not keep track of the amount of players. This makes moving along the queue without interrupting the game possible. When players move along the queue they also control a different position of the wave. When a new player enters the camera FOV he/she (usually) controls the part of the wave right after where the object enters the screen. Bearing in mind that new players can join at any time and since the goal of the game is to score combos we do not want the presence of a new players to be linked to a high chance that the combo will be interrupted. This is why players near the edges have little effect on the game, and targets or obstacles that can break the combo do not reach this far.

Figure : Simplistic overview of themed game being played.

Finally, also with respect to queue dynamics, players can also leave at anytime, so that when the queue moves along and the last player goes forward in the queue and leaves the camera’s FOV, the player before him/her will simply move along and replace the player that just left.

The game always plays in the same mode, from which different play states can be reached depending on the players’ competency to maintain the combo.

# References

1. Jmonkey engine 3.0  http://jmonkeyengine.org/
2. Open-cv 2.4.10 for Java http://opencv.org/