

pyWuzzler v0.7



Developer:

Benjamin Gmeiner, BSc. benjamin.gmeiner@fh-hagenberg.at

Yen-Chia Lin, BSc. yen-chia.lin@fh-hagenberg.at

Supervisor:

Andreas Jakl, MSc. andreas.jakl@fh-hagenberg.at

Students at:

University of Applied Sciences Hagenberg
Master's Degree Mobile Computing

Website:

<http://www.fh-ooe.at/mc>

YouTube Video:

<http://www.youtube.com/watch?v=jzvceJAgpX8>

1 Introduction

The idea of **pyWuzzler** is to use the **acceleration sensor** of the mobile phone Nokia N95 to create a table soccer game. As a whole game might be difficult to play on such a small display, the game play is reduced to a head to head with the goalkeeper, where the player can control the players of the attacking row. The goalkeeper is controlled by an AI to make it more difficult to score a goal. The **main goal** of the game is to score as many goals as possible in two minutes.

The game is currently in an early version of development, so there might be a few bugs and performance problems. It is currently available only for the Symbian OS S60 3.x platform with C++ / Python installed and will work only on S60 3rd devices with integrated accelerometer (tested only with Nokia N95).

2 Features

The main feature of the game is the use of the **accelerometer** of the phone to control the attacking players. This new way of interacting with a mobile game was part of the course “Interaktionstechnologien” during which this game was developed. The course was hold by Jakl Andreas, assistant professor at the University of Applied Sciences Hagenberg.

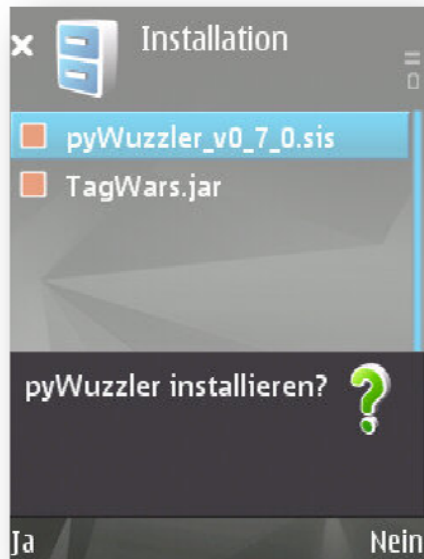
Attacking players (red players) can be moved through sidewise change of the **inclination** of the mobile phone. A shot can be triggered by fast **turning** N95 (see below section “How to play”). Goalkeeper (blue player) is controlled by an **AI** to prevent from the player to shoot too many goals.

In addition **realistic ball physics** was implemented into the game. Therefore the ball will bounce off objects such as the wall, players and goalkeeper. The speed of the ball will be reduced through bouncing off objects and the ball friction. The ball position will be reset once the ball speed falls below a certain threshold or rolls back behind the attacking players.

At the beginning the ball physics is **simplified**. The ball can only be moved sidewise to make it easier for the player to pass the ball between the attacking players. So it won't lose the ball so easily. The ball can only move in all directions and bounce off other objects after a shoot is performed.

3 Installation

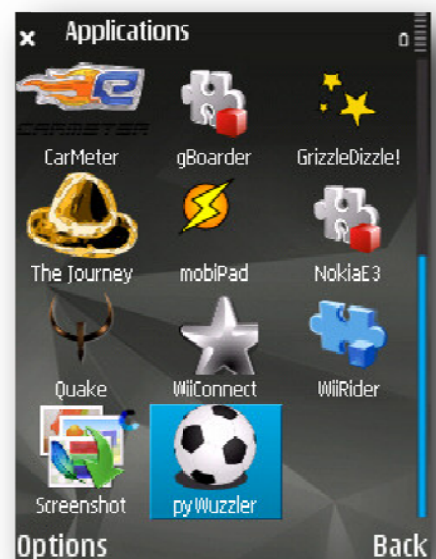
1. First make sure **PyS60** is installed on the mobile phone (<http://sourceforge.net/projects/pys60>) During development PyS60 v1.4.2 was used. Should also work with later versions.
2. Copy "pyWuzzler_v0_7_0.sis" onto the Nokia N95
3. Open "pyWuzzler_v0_7_0.sis" on the phone to install the game. pyWuzzler must be installed on the **same** drive as PyS60 (C: for internal memory or E: for external memory) otherwise the game won't start



4. Follow the instructions on the screen to complete the installation.

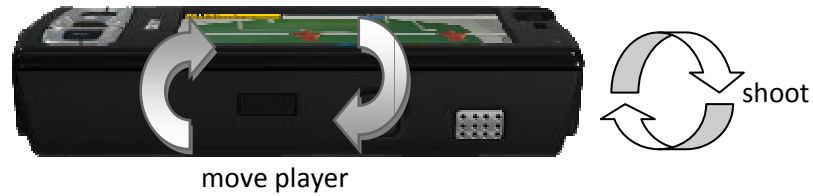


5. Start the game by starting "pyWuzzler" in the Applications.



4 Game instructions

The built-in accelerometer from Nokia N95 phone is used to control the game. To see how the controls work in live please see the YouTube video at <http://www.youtube.com/watch?v=jzvceJAgpX8>.



Moving attack players:

Change the **inclination** of the mobile phone to move the attacking players to change their position.

Performing a shoot:

Fast **turning** of the mobile phone will trigger a shot.

Key bindings:

“2” → Adjust the neutral position of the attacking player. Hold the mobile phone horizontally and turn it so you can see the game screen clearly. Now press “2” on the mobile phone to use the new physical position as the neutral position in the game.

“5” → Resets the ball to its original position.

“*” → Enables/disables the sound.

5 Screenshots

Initialising.....
> starting initialisation (run hamster)
> initialising basic properties
> loading sounds
> loading player models
> loading ball sprites and creating mask
> loading pole sprites and creating mask
> loading background

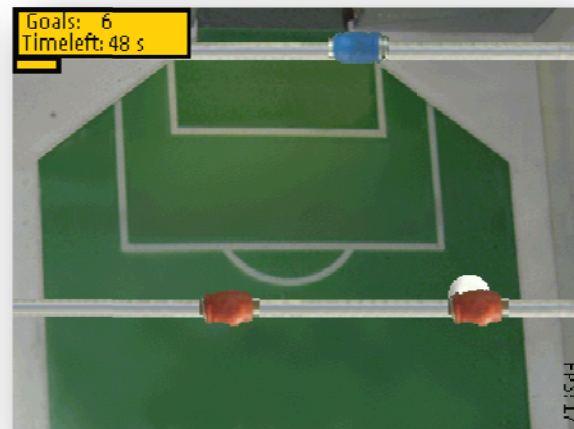
Initialising.....
> starting initialisation (run hamster)
> initialising basic properties
> loading sounds
> loading player models
> loading ball sprites and creating mask
> loading pole sprites and creating mask
> loading background
> initialising some properties, we're almost

Start by pressing OK



OK

Abbruch



OK

Abbruch