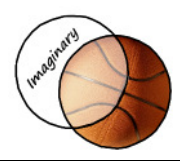


# Imaginary Basketball (Project WS 2010)

---

## Task

	Project onto the floor, track players' hands using optitrack, let them pass and throw an imaginary ball.
---	--

## People

- Emilia Wittmers [<http://www.emilia-wittmers.de>]: I like the idea of implementing a virtual basketball field because I'm a basketball player and love this sport. I visited the lecture 'Designing Interactive Systems' and designed PaintyFeet (a touchfloor application for foot painting).
- Sven Wagner-Boysen (master, 3rd): I generally like the idea of getting more movement actions into virtual sports games. Bring-in prototyping experiences from ME310 course and programming skills in Java and C#.

## Project

		<p><b>Interactive Collaboration via Shared Illusions</b></p> <p>Is it possible to combine human computer action with the imaginary world of collaborating people? How does interactive collaboration works if people play with things - like a ball - they don't see? What could be possible when people interact over fictions that is not possible in the real world?</p> <p>A joystick was yesterday, the Wii Nunchuks are today, but shared illusions are tomorrow!</p>
--	--	---

## ImaginaryBasketball

ImaginaryBasketball is played by at least two people wearing a glove on the right or left hand which has several retroreflective markers on it shaping a unique pattern to identify the players by a camera system.

The goal of each player is to score more points than the others before the game ends after a defined time interval. To get a point the player has to throw the imaginary ball into the direction of the imaginary basket. If another player catches the ball before it hits the basket it is his turn to score. Collides the ball with the wall or is thrown into the basket it falls onto the floor and has to be picked up from a player. A player who scored recently can't pick up the ball; he has to steal it from the player who picks it up. This player can steal the ball by catching it or when the ball-owning-player begins to dribble.

## Milestones

PASSWORD FOR ALL VIDEOS IS: **ImBall**

<p>- MISSING VIDEO PREVIEW -</p>	<p><b>11.November - First Presentation</b> pdf-slides and odp- &amp; ppt-slides + videos</p>
<p>- MISSING VIDEO PREVIEW -</p>	<p><b>Week 3</b> - recognition of holdUp and throw - if player1 throws the ball and player2 is holding up his hand, player2 gets the ball (wherever he is standing)</p>
<p>- MISSING VIDEO PREVIEW -</p>	<p><b>Week 4</b> - soundoutput</p>

- MISSING VIDEO PREVIEW -

**Week 5**

- direction recognition: could a player catch the ball if I throw it in a specific direction? (works not exactly at this moment)

- MISSING VIDEO PREVIEW -

**Week 6**

- correct direction recognition  
- more than two players can play together  
- if the player, who has the ball, throws the ball, another player catches the ball only if he holds his hand up in the right direction

- MISSING VIDEO PREVIEW -

**Week 7**

- the ball has speed  
- a player can intercept the flying ball  
- when the ball is thrown, it can fall at the floor if there is no catchable hand identified in the right moment  
a player can pick up the ball from the floor



**Week 8**

- recognition of dribbling
- the ball owning player can dribble
- if the ball owning player dribbles, another player can steal the ball from him

- MISSING VIDEO PREVIEW -

**Week 9**

- user observation

- MISSING VIDEO PREVIEW -

**Week 10**

- there is a defined basket direction
- if a player throws the ball into the direction of the basket and there is no player who intercepts the ball then the throwing player gets a point
- the score is displayed on the console
- after a defined interval of time the game finishes and the winner(s) will be nominated

- MISSING VIDEO PREVIEW -

27.January - Final Presentation

final.pdf and final.ppt

## Results of the User Observation

We observed people while they were playing ImaginaryBasketball.

Our testpersons had to figure out how to pick up an imaginary ball and how to throw it:

- The pickUp-gesture recognition works very well. The testpersons picked up the imaginary ball from the floor without any problems.
- To throw the imaginary ball was more difficult. One person threw the ball like he wants to throw it into the basket - that gesture was well recognized. After we explained how the throw-gesture works it was better identified by the system than before.
- The testpersons figured out - by their own - that they can dribble and how it works.

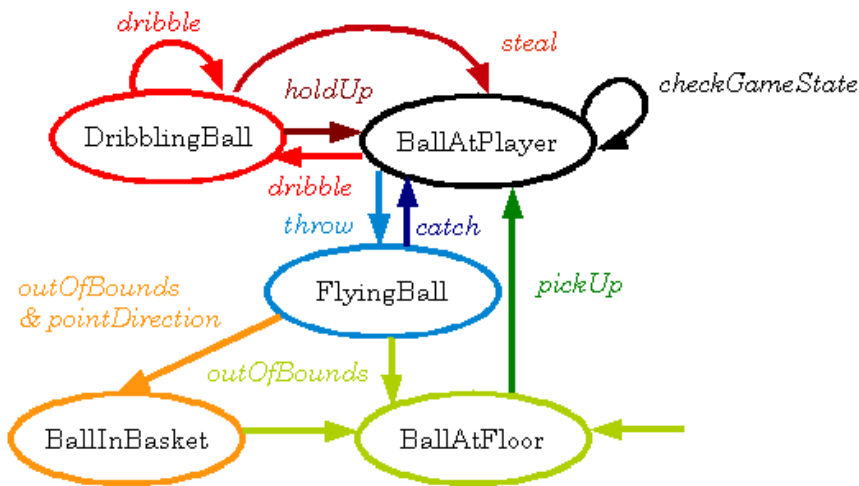
The testers had very much fun by testing our application. They liked playing with a ball they can't see.

## How it works

### The System behind the Game



### Game Engine



#### BallAtFloor

1. At the beginning of the game the imaginary ball lies on the floor.
2. If a player picks up the ball (see Gestures.PickUp), he becomes the ball-owning-player and the game state changes from "BallAtFloor" into "BallAtPlayer".
3. The game state returns into "BallAtFloor" if a player throws a ball and there is no other player who catches the ball (see Game\_Engine.FlyingBall and Game\_Engine.BallInBasket).

#### BallAtPlayer

1. The game state changes from "BallAtFloor" into "BallAtPlayer" if a player picks up the ball from the floor (see Game\_Engine.BallAtFloor).
2. A player who owns the imaginary ball can throw it (see Gestures.Throw) or dribble (see Gestures.Dribbling). If the player throws the ball the game state changes from "BallAtPlayer" into "FlyingBall" (see Game\_Engine.FlyingBall). If the player dribbles the ball the game state changes from "BallAtPlayer" into "DribblingBall" (see Game\_Engine.DribblingBall).
3. The game state returns from "FlyingBall" into "BallAtPlayer" if there is another player who catches the ball after it has been thrown (see Game\_Engine.FlyingBall).
4. The game state returns from "DribblingBall" into "BallAtPlayer" if there is another player who steals the ball (see Game\_Engine.DribblingBall) or if the dribbling player holds his hand up to throw the ball (see Gestures.HoldUp and Gestures.Throw).
5. When the current game state is "BallAtPlayer" the players can check who the ball-owning-player is if one of them holds his hand up (see Gestures.HoldUp) and the others hold their hand down like they want to do a pickUp-gesture (see Gestures.PickUp). If the players check the game state as described the system says the name of the ball-owning-player.

#### FlyingBall

1. The games state changes from "BallAtPlayer" into "FlyingBall" when the ball-owning-player throws the ball (see Gestures.Throw).
2. When the game state changes into "FlyingBall" the StartFlightPosition and the StartFlightTime is set. If there is no player who could catch the flying imaginary ball before it reaches the limit of a defined distance the game state will change from "BallAtPlayer" into "BallInBasket" (if the ball has been thrown into the direction of the basket - see GameEngine.BallInBasket and Gestures.PointDirectionCondition) or into "BallAtFloor" (if the ball has not been thrown into the direction of the basket and therefore it crashes to room's wall - see BallAtFloor). If there are players who could catch the ball (see Gestures.Catchable) the best fitting player (= the player who has to get the ball) is calculated by the valuation system out of normed angles and distances whereby the distance is slightly more weighted. The game state will change into "BallAtPlayer".

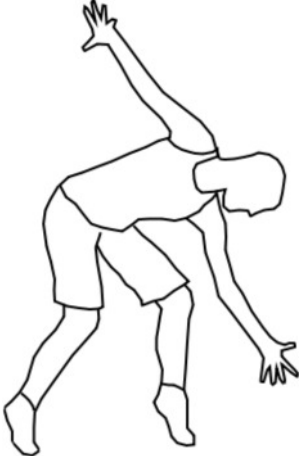


#### BallInBasket




1. The game state changes from "FlyingBall" into "BallInBasket" if the flying imaginary ball reaches the limit of a defined distance before a player could catch it and the ball-owning-player threw the ball into the direction of the basket (see Gestures.PointDirectionCondition).
2. Is the current game state "BallInBasket" the player who threw the ball gets a point and (to make the game a bit more interesting) is blocked for the next pick up gesture (see Gesures.PickUp). The game state turns automatically in the "BallAtFloor" state.

## DribblingBall

1. The game state changes from "BallAtPlayer" into "DribblingBall" if the player begins to dribble (see Gestures.Dribble).
2. Is the current game state "DribblingBall" another player could steal the ball from the ball-owning-player (see Gestures.CouldSteal) and the game state will change into "BallAtPlayer".
3. The game state will also change from "DribblingBall" into "BallAtPlayer" if the ball-owning-player holds his hand up (see Gestures.HoldUp) to throw the ball (see Gestures.Throw).

## Gesture Recognition

	<p><b>PickUp</b></p> <p>The pickUp-gesture is needed to pick up the imaginary ball from the floor or when the players want to know who the ball-owning-player is. It is recognized by the following constraints:</p> <ol style="list-style-type: none"><li>1. the angle between the normal vector of the room's y-axis and the normal vector of the hand has to be bigger than 90 degree and smaller than 120 degree</li><li>2. the hand's x-orientation has to be negative</li></ol>
	<p><b>HoldUp</b></p> <p>The holdUp-gesture is needed to recognize the throw-gesture or when the players want to know who the ball-owning-player is. It is recognized by the following constraints:</p> <ol style="list-style-type: none"><li>1. the angle between the normal vector of the room's y-axis and the normal vector of the hand has to be bigger than 90 degree and smaller than 120 degree</li><li>2. the hand's x-orientation has to be bigger than 70 degree and smaller than 120 degree</li></ol>
	<p><b>Throw</b></p> <p>The throw-gesture is needed to identify a shot. It is recognized by the following constraints:</p> <ol style="list-style-type: none"><li>1. the last gesture has to be a holdUp-gesture</li><li>2. the elapsed time between the holdUp-gesture recognition has to be smaller than 800 milliseconds</li><li>3. the angle between the normal vector of the room's y-axis and the current normal vector of the hand has to be smaller than 25 degree</li></ol>

	<p><b>Catch</b></p> <p>The catch-gesture is needed to identify if a player could catch the imaginary ball. A player could catch the ball if the following constraints are fulfilled:</p> <ol style="list-style-type: none"> <li>1. the angle between the normal vector of the ball-owning-player's hand and the normal vector of the hand which is checked to be "catchable" has to be bigger than 100 degree and smaller than 120 degree</li> <li>2. the angle between the normal vector of the ball-owning-player's hand and the vector between the position of the hand which is checked to be "catchable" and the position of the ball-owning-player's hand when it made the holdUp-gesture has to be smaller than 30 degree</li> </ol> <p>If there a more than one player who could catch the ball, the player gets the ball whose hand performs better in direction and orientation to the ball-owning-player's hand at the holdUp-gesture.</p>
	<p><b>Dribble</b></p> <p>The dribble.gesture is needed for dribbling. It is recognized by the following constraints:</p> <ol style="list-style-type: none"> <li>1. there has to be a holdStraight-gesture been recognized (the angle between the normal vector of the room's y-axis and the normal vector of the hand has to be bigger than 5 degree and smaller than 30 degree)</li> <li>2. after the first holdStraight-gesture recognition has to be a raise-gesture been recognized in the time interval of 700 milliseconds (the angle between the normal vector of the room's y-axis and the normal vector of the hand has to be bigger than 30 degree)</li> <li>3. after the raise-gesture recognition has to be a second holdStraight-gesture been recognized in the time interval of 800 milliseconds</li> </ol>
	<p><b>Steal</b></p> <p>The steal-gesture is needed when a player wants to steal the imaginary ball from the dribbling ball-owning-player. A player could steal the ball if the following constraints are fulfilled:</p> <ol style="list-style-type: none"> <li>1. the current x-value of the hand of the player who wants to steal the ball has to lie in the intervall [x-position of the ball-owning-player's hand minus 10 centimeter;x-position of the ball-owning-player's hand plus 10 centimeter]</li> <li>2. the current z-value of the hand of the player who wants to steal the ball has to lie in the intervall [z-position of the ball-owning-player's hand minus 10 centimeter;z-position of the ball-owning-player's hand plus 10 centimeter]</li> <li>3. the current y-value of the hand of the player who wants to steal the ball has to be smaller than the current y-value of the ball-owning-player's hand</li> </ol>

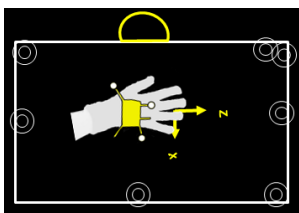
## How to Install

### Requirements

- Visual Studio 2010
- .Net 4.0
- Source code on Q: \\fs3\projekte\$\Baudisch\Projects\ImaginaryBall\SourceCode.zip

### Create Trackables with Tracking Tools

- be sure that you register your rigid bodies in the correct direction relative to your room's coordinate system
- the correct position/ direction is shown in the figure below



### Getting Started

- Set up Visual Studio and load the Imaginary Ball solution file.
- Check if all library dependencies are resolved. The VRPN .Net implementation is the only external Lib we are using. (path: TrackingSpike\Libraries\VrpnNet.dll). All others are included in .Net 4.0
- Start the VRPN-Streaming Engine from the Tracking Tools Software of the Optitrack-System
- Apply the right configuration settings in /TrackingSpike/Configuration/ImBallConfiguration.cs
  - VRPN Streaming Engine IP
  - Mapping player names to Rigid-Body names (the tracked Hands)
  - Game-Settings (Duration, Floor expansion, Ball Speed, ...)
- Run the application

## Implementation Overview

- Tracking(Tracking/Tracker.cs | Tracking/TrackerDataManager.cs)
  - All VRPN-Events arrive at the Tracker
    - Hand position is updated
    - orientation quaternion is transformed to a 3D-Vector describing (pitch, roll, yaw)
- The TrackerDataManager provides capabilities for storing and redirecting tracking events
  - Possible configurations
    - Receiving tracking data from the Tracker and and redirecting to GRM
    - Writing to file
    - Reading from file and redirection to GRM
- Gesture Recognition (GestureRecognitionMachine/GestureRecognitionMachine.cs)
  - Receives the preprocessed Tracking Events (#ReceiveTrackingData)
  - Then performs the gesture recognition
  - And sends the gestures changing the game state to the game engine (#SendGestureData)
- Game Engine (/GameLogic/GameEngine.cs)
  - Implemented as simple state machine
  - Event processing see: #ProcessGestureEvent -> #HandleGameEvent
  - Each state implements a Handle method defining the transition conditions

## Another Approach: Quantum Game

