HORIZON 2020 FRAMEWORK PROGRAMME
ICT – 21 -2014
Advanced digital gaming/gamification technologies

Gamification of Prosocial Learning for Increased Youth Inclusion and Academic Achievement

D4.6
2nd Natural Game Interactions
This document describes the natural user interaction approaches used in the prosocial games of the project.

Author(s)  Konstantinos Apostolakis (CERTH), Kosmas Dimitropoulos (CERTH)
Contributor(s)  Athanasios Psaltis (CERTH), Petros Daras (CERTH)
Reviewer(s)  Kam Star (PG), Erik Robertson (RK)

Dissemination level  
☑️ public
☐ internal
☐ confidential
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>VGB</td>
<td>Visual Gesture Builder</td>
</tr>
<tr>
<td>GCI</td>
<td>Gesture Controlled Interface</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Models</td>
</tr>
<tr>
<td>NUI</td>
<td>Natural User Interface</td>
</tr>
<tr>
<td>NGI</td>
<td>Natural Game Interactions</td>
</tr>
<tr>
<td>PoT</td>
<td>Path of Trust (game)</td>
</tr>
<tr>
<td>PL</td>
<td>ProsocialLearn</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit</td>
</tr>
</tbody>
</table>
Executive summary

The current deliverable is one of the outputs of work package **WP4: Dynamic and Personalized Game Elements for Prosocial Learning**. It is a public document focusing on the **Deliverable 4.6: 2nd Natural Game Interactions**. This document will be made available on the project website for external parties interested in techniques and modules for natural interaction in games. This is the second version describing a mid-air-based –gestures-based action recognition API (Natural Game Interactions – NGI API) that will provide game developers with controllers for gesture recognition using Microsoft Kinect v2 sensor. The primary aim taken into consideration in the design process of NGI API is to provide easy-to-use tools supporting NGI to the future developers who will construct educational games using the ProsocialLearn Platform. Towards this end, a variety of capturing tools and NGI modules are presented.
Table of Contents
1 Introduction ..................................................................................................................................... 8
   1.1 Overall WP4 structure ........................................................................................................ 8
   1.2 Positioning of Task 4.3 and purpose of this deliverable .................................................. 8
   1.3 Structure of the Document ............................................................................................. 8
2 ProsocialLearn gesture-based interaction system ........................................................................ 9
   2.1 Implementation and evaluation ...................................................................................... 9
3 ProsocialLearn NGI API ........................................................................................................... 15
   3.1 NGI Module 1: The module for predefined supported gestures .................................. 15
   3.2 NGI Module 2: The module for the insertion of new gestures ..................................... 19
   3.3 Gesture Database .......................................................................................................... 20
   3.4 Gesture recording platform ........................................................................................... 24
   3.5 Visual Gesture Builder ................................................................................................... 26
4 Conclusions .................................................................................................................................. 33
5 References ................................................................................................................................... 34
List of Figures

Figure 1. Sample data captured by the Kinect for Xbox 360 sensor ......................................................... 11
Figure 2. *Path of Trust DX* and *Path of Trust: The Christmas Edition* title screens. ................................. 12
Figure 3. Junction and Corridor tile types making up the Path of Trust map maze................................. 12
Figure 4. PoT architecture diagram. ........................................................................................................... 13
Figure 5. Controlling the Guide character using Kinect body gestures..................................................... 13
Figure 6. Controlling the Muscle character using Kinect body gestures................................................... 14
Figure 7. Body gestures for controlling player Muscle/Guide actions using Kinect..................................... 14
Figure 8. Block diagram of NGI module 1................................................................................................. 15
Figure 9. Joints provided by Kinect 2 SDK ............................................................................................... 16
Figure 10. Hidden Markov Model. ........................................................................................................... 17
Figure 11. Block diagram of NGI module 2............................................................................................... 19
Figure 12. Direction gestures .................................................................................................................... 21
Figure 13. Fly, jump, squat and lean gestures to avoid virtual obstacles ................................................ 22
Figure 14. Throw gesture to manipulate virtual objects ........................................................................... 23
Figure 15. Exergame gestures ................................................................................................................ 24
Figure 16. Connect to service button....................................................................................................... 24
Figure 17. Kinect Studio Stream selection menu ...................................................................................... 25
Figure 18. Record, pause, stop and Playback. .......................................................................................... 25
Figure 19. VGB menu, open new solution................................................................................................. 27
Figure 20. Building gesture dataset ....................................................................................................... 31
Figure 21. Previewing arm left, fly and lean right gestures ...................................................................... 32
Figure 22. VGB process............................................................................................................................ 32
List of Tables

Table 1: NGI ProsocialLearn gesture database................................................................. 20
1 Introduction

This section provides detailed information about the purpose of WP4 in general, placing of this Task and accompanying deliverable, as well as the scope and structure of the document, which set the tone for the intended audience and interested readers.

1.1 Overall WP4 structure

The aim of WP4 is to develop and enhance leisure game and related technologies so that they can be used for prosocial game design and development. Tasks within this WP will select, enhance, configure and fine-tune game technologies considering input handling, game mechanics and graphical elements. The enhancements will focus on supporting games of prosocial character, and adaptive and personalized interactions to optimize student learning potential. The components developed in WP4 including artificial intelligence algorithms for personalized prosocial games supporting offline and online adaptation, collection of game mechanics for prosocial skills acquisition, game interface controllers support natural game interaction based on gestures and virtual characters, capable of conveying prosocial messages, will be delivered to WP5 for integration into the ProsocialLearn Platform.

This document will focus on natural game interactions based on gesture controlled interfaces (GCI), presenting in detail the API for natural game interaction that will be incorporated in ProsocialLearn Platform.

1.2 Positioning of Task 4.3 and purpose of this deliverable

The current deliverable is the continuation (second and final iteration) of D4.5 for Task 4.3, which aims at defining and describing the components of NGI API that will be incorporated in ProsocialLearn Platform. The 1st version (D4.5) was delivered at M18 and provided a detailed introduction concerning natural game interactions, building a strong background that will lead to the selection of specific techniques and approaches for the design of NGI ProsocialLearn system. In this deliverable, we will describe an easy-to-use Kinect-based gesture capturing tool, to be used by developers for incorporating their own specific gesture-driven interactions with their game content. A step by step guide will be presented, while special emphasis will be given on how to build a gesture detection mechanism. Furthermore, a new small-scale gesture dataset will be introduced, which involves a total amount of 20 subjects performing game-related gestures.

1.3 Structure of the Document

This document contains the following key sections, conveniently detailed in the list below:

- **Section 1: Introduction** – an introductory section, i.e. this present section, which describes the WP as a whole, as well as the main purpose of the Task that generated this document.
- **Section 1 Error! No se encuentra el origen de la referencia.: ProsocialLearn gesture-based interaction system** – the aim of this Section is to present initial developments that will ensure that game interactions modules are available and robust for integration into ProsocialLearn NGI system.
- **Section 1 Error! No se encuentra el origen de la referencia.: ProsocialLearn NGI API** – in this section, a detailed description of our NGI API will be presented.
- **Section 4: Conclusion** – this section presents the conclusion of the document.
2 ProsocialLearn gesture-based interaction system

In D4.5 we provided a brief review of gesture-based interactions techniques and algorithms for gesture recognition. In this document we will present the NGI modules that we have designed, implemented and integrated on the Path of Trust (PoT) game, emphasizing on real-time interaction. Subsequently, based on the conclusions derived from experiments and previous analysis, we will present the overall architecture and functionality of the gesture-based interaction system that will be incorporated in PL platform.

2.1 Implementation and evaluation

In the design of NGI modules for PL prototype games, we had to take into consideration the following requirements:

- A set of interactions for gameplay scenarios should be selected.
- These interactions should feel natural, intuitive for children aged 7-10 in order to enhance game experience.
- The sensing technologies should be non-intrusive.
- The sensors should be low-cost and portable, so that they could be used for gameplay in the classroom or at home.
- The system should recognize with accuracy the users’ interactions in real time, meaning that any processing and computation should be suitable for real time applications and especially game applications that normally require a lot of processing for graphics and for AI algorithms.

Taking into account the aforementioned requirements, the first step we had to do was the selection of the type of interactions. In previous deliverable D4.5, we have provided an overview of gesture-based interactions, presenting both advantages and possible limitations and problems encountered in the utilization of such kind of systems. Due to intuitiveness and ease of use (requirements of great importance when dealing with the design of games aiming at young children), we selected body as the natural interaction alternative to traditional interactions for ProsocialLearn games.

The next step was to select the sensing technologies for capturing effectively the body gestures. The commercial sensor that meets these requirements is Microsoft Kinect. Kinect constitutes the predominant device used for natural interactions since it was launched into the consumer’s market in 2010. Of course, the performance of these sensors needs testing and validation, something that happened through a series of small-scale experiments. Another advantage that novel sensing devices such as Kinect presents versus conventional cameras or 3D cameras, is the incorporation of skeleton-tracking, image segmentation and preprocessing algorithms in the software that accompanies the hardware. Providing via their SDKs classes and data, almost ready for feed in recognition algorithms, they take a great burden off the developers’ shoulders.

Having chosen the capturing device, we needed to determine the appropriate methods for accurate gesture recognition including feature selection and gesture classification. The recognition algorithms that we have presented in detail in D4.5 are the ones that seem to be suitable for the recognition tasks in the ProsocialLearn project.
2.1.1 A brief overview of input devices

As we mentioned, Kinect has been selected as input device. Below, we will provide some information about the functionality, utilization and development of these sensors.

2.1.1.1 Kinect

Late in 2010, Microsoft attempted to revolutionize the way game players interacted with their games and gaming hardware (Xbox 360) by introducing Kinect sensor, the first in a line of motion sensing input devices intended to replace the traditional controller. The original device featured an RGB camera, a depth sensor and a multi-array microphone running proprietary software, and builds on software technology developed internally and range camera technology developed by the now defunct Israeli company PrimeSense. Instantly, the device broke the Guinness World Record for fastest selling consumer electronics device, and was positively accepted by the homebrew software and scientific community (Zhang, 2012). In 2012, Microsoft introduced a Windows version of the device along with an official SDK to provide developers with the Kinect capabilities in order to build applications with C++, C# and Visual Basic using Microsoft Visual Studio 2010 and beyond. An upgraded version marketed with the company’s latest gaming hardware (the Xbox One2) was released late in 2013.

Since the original Kinect for Xbox 360 sensor broke ground in 2010, several research topics surrounding low-cost motion capturing systems emerged (Berger et al, 2011). More specifically, research work capitalized upon full-body skeleton tracking capabilities offered for the device through a number of software development kits created for the sensor, most notably by Kinect project partner PrimeSense (OpenNI), until the release of the official SDK by Microsoft (Shotton et al, 2012). Shortly, research works on rehabilitation based on bodily activity patterns during gameplay emerged, capitalizing on the sensor’s low cost and firmly established affiliation with the games industry (Chang et al, 2011) (Lange et al, 2011).

In the ProsocialLearn platform, we utilize Kinect both for the extraction of features related to the players’ 3D full body motion and for game interactions. In this respect, we monitor and process data incoming from every visual component provided by, and the SDKs built for the sensor. Figure 1. Sample data captured by the Kinect for Xbox 360 sensor. Left image (a) shows RGB input, while the right image (b) shows depth and skeleton tracking results obtained from the OpenNI library demonstrates the raw data input of our feature extraction algorithms for both sensors.

---


2.1.1.2 NUI controllers at Path of Trust Game

Path of Trust (PoT) is a two-player (Apostolakis et al, 2015), endless running maze game in which players strive to collect treasure while trying to avoid mummies and traps. One player assumes the role of the Guide (Figure 5), a character assumed to have explored the maze before and therefore, able to navigate through the corridors via a top-down view of the dungeon map. The other player is put in control of the endless running Muscle character (Figure 6), which is responsible to navigate the maze, but has no information on the layout, of the maze or the room contents. As the ancient treasure is guarded by terrifying mummies who dwell in the dungeon corridors, the two unlikely partners need to form a trusting bond in order to fulfill their mission, as the Muscle needs to trust his partner to provide guidance away from danger, while the Guide must trust their partner to listen to follow his directions. The duo navigates the maze by having the Muscle carry the Guide on his back. During each game cycle, the characters find themselves in one of the maze’s many rooms, which end up in junctions leading up to 3 different directions. Before entering the room, the Guide has already been shown a short glimpse of the contents in each of the adjacent rooms and has to pick a decision for the Muscle character to follow. The Muscle then gets a small time window to decide whether to actually heed the Guide’s advice and proceeds to take one of the available routes. Afterwards, the Muscle gets a short time to either collect the treasure or avoid a hazard running inside a small corridor leading up to the next room/junction while the Guide is again shown the contents of the adjacent rooms there, from which point the cycle repeats.
A similar setting was established in a Christmas Edition (PoT Xmas) of the game, included as an unlockable theme during the months November to January. In this version, the characters are replaced by Santa’s elves following instructions on collecting toys for Santa’s sleigh, while the mummies have been replaced by the Grinch, attempting to steal Christmas. Christmas Edition is one-player, since the Guide is represented by AI-driven NPC. Also, Path of Trust DX edition (PoT DX) is revised one-player edition of PoT with improved graphics and AI-driven Guide character. The two latter versions’ title screens can be seen side by side in Figure 2.

![Figure 2. Path of Trust DX and Path of Trust: The Christmas Edition title screens.](image)

In figure 4, it is depicted the PoT server-client architecture diagram and the flow of game data. As we can see, the main task of a PoT client is to provide a generic framework for input device communication. Each device constitutes a child thread of the game client process, while all interactions are carried out through the use of local sockets. Messaging with each device follows a unified protocol. Preserving a synchronization scheme with the game server, the client initializes...
each sensor and exchanges game information between the sensors and the game. A web browser module is responsible for executing game actions and displaying the game world to the users. This system architecture allows easy extension of the PoT to new sensors, thus making the game independent from any specific input device.

Figure 4. PoT architecture diagram.

As stated earlier in this Section, PoT can be played either using a standard interaction interface such as a keyboard to control actions via the arrow keys, or with an NGI interface that uses the user’s bodily gestures as an input controller to enhance player engagement. PoT gesture recognition interface supports both versions of the Microsoft Kinect sensor for full body motion tracking. Gesture recognition using the Kinect is done by tracking skeletal joints on the players’ hands, using our NGI
module to determine whether a valid gesture is performed. In order to simplify procedures for students playing the game at this young age (7-10), simple directional gestures were incorporated in the system: players extend their left or right arm in order to either declare the intention to suggest/turn towards the desired direction when in a crossroads section, or navigate their run towards the left or right whenever inside a corridor section. In order to declare an intention to go forward, players are asked to extend both arms forward (Figure 7).

![Figure 6. Controlling the Muscle character using Kinect body gestures.](image)

![Figure 7. Body gestures for controlling player Muscle/Guide actions using Kinect.](image)
3 **ProsocialLearn NGI API**

Having gained a valuable insight from the small-scale experiments, have implemented a mid-air-based -gestures-based action recognition API (NGI API) that will provide game developers with controllers for gesture recognition using Microsoft Kinect v2 sensor. However, in the future we aim to focus mainly on Kinect 2, since it proved to be more stable during the first small-scale experiments. The primary aim taken into consideration in the design process of NGI API was to provide easy-to-use tools supporting NGI to the future developers who will construct educational games using the ProsocialLearn Platform.

The NGI API consists of two different modules:

### 3.1 NGI Module 1: The module for predefined supported gestures

It is of great importance to provide game developers the necessary support for the recognition of a standard set of gestures/actions that have been proven intuitive for children gameplay. This gesture repertoire could include navigation gestures, jump gestures to avoid virtual obstacles, grasp and throw gestures to manipulate virtual objects, exergame gestures. The training for these actions/gestures will be done with Hidden Markov Models and the module will provide the functions for the recognition of these actions/gestures. The block diagram of NGI module 1 is provided in Figure 8.

**Figure 8. Block diagram of NGI module 1**

Kinect 2 SDK provides the positions and orientation angles of 25 skeleton joints as we can see in Figure 9, at every frame. This information can be used either directly as input in recognition algorithms or alternatively, it can be processed to give a set of higher level features that could act as descriptor for body gestures.
A more advanced approach for gesture recognition with Kinect is the use of a Hidden Markov Model (HMM)-based recognition method, briefly described below:

The input for the method is the depth data which is collected for the Kinect v2 sensor. A skeleton tracking algorithm is used for the continuous detection of the joints (25) in human body as shown in Figure 9. For the identification of similar action, spherical angles between particular joints (wrists, elbows, knees, and feet) are measured. At the end, HMM (Hidden Markov Model) performs the action recognition. As we mentioned the initial step in action analysis process is the pose estimation method. This step is applied to make the process invariant to differences in appearance and body shapes. All angles are measured using the Torso joint as a reference. The proposed action representation is computed by the use of only a subset of the supported joints. We employ a set of HMMs, where we have an individual HMM for every supported action/gesture.

A brief overview of the proposed module:

- Use of front-view Kinect depth map
- Application of human skeleton tracking algorithm
- Body pose estimation from detected joints
  - Definition of new coordinate system per frame
- Selection of particular joints for realizing action recognition
  - Selected joints: wrists, elbows, knees, feet
- Estimation of vectors connecting the torso with every selected joint
  - Computation of the 3 angles of the formed vectors
- Hidden Markov Models used for action recognition
3.1.1 A brief overview of Hidden Markov Models

In human activity recognition and gesture recognition as well, the most accepted classification techniques are Hidden Markov Models (HMMs), because activities have temporal structure and HMMs can tackle with this kind of processes. However, others methods have been successfully used: Neural Networks, Neuro-Fuzzy systems, C4.5 algorithm. etc. (Piccardi & Molina, 2007).

Gesture sequences often present a complex underlying temporal structure and models that incorporate hidden structures have proven to be advantageous for recognition tasks. Most existing approaches to gesture recognition with hidden states employ a Hidden Markov Model (HMM) or a suitable variant (e.g. a factored or coupled state model) to model gesture streams. Below, we will briefly introduce HMM and briefly examine their functionality.

Be a system that can be described using a set of N different states, where random transitions are produced over time, according to a given probability distribution for each state. The state on the system on each moment depends on the state that it was in the previous moments. This kind of stochastic process is called “Markov Model”. Additionally, if the present state of the system cannot be observed, i.e., it could be only measured by an effect that it produces, the system is called “Hidden Markov Model”.

This in mathematics terms means that HMM is a generative probabilistic model which is used for generating hidden states from observable data. The main goal of the model is to determine the hidden state sequence \( x_1x_2...x_T \) that corresponds to the output sequence of observations \( y_1y_2...y_T \). Another important goal is to learn model parameters reliably from the history of observed output sequences. Figure 10 shows a graphical representation of an HMM.

![Figure 10. Hidden Markov Model](image)

A Hidden Markov Model \( \lambda \) (Rabiner, 1989) is defined by the tuple \( \lambda \equiv (S, M, A, B, \pi) \) where:

- \( S = \{ S_0, ... , S_N \} \) is the set of hidden states of the model. The state at time \( t \) is denoted by \( x_t \).
- \( M \) is the set of observation symbols.
\[ A = \{a_{ij}\} \] is the state transition probability (transitional probability) distribution:
\[
a_{ij} = p(x_{t+1} = S_j|x_t = S_i)
\]
\[ B = \{b_j\} \] is the observation symbol probability distribution (emission probability) in state \( j \):
\[
b_j = p(u|x_t = j)
\]
\[ \Pi = \{\pi_j\} \] is the initial state probability distribution:
\[
\pi_j = p(x_0 = j)
\]

As we mentioned while introducing HMM, the model requires two independence assumptions for tractable inference:

- The 1\textsuperscript{st} order Markov assumption of transition: \( P(x_t|x_1, x_2, \ldots, x_{t-1}) = P(x_t|x_{t-1}) \) (5), meaning that the future state depends only on the current state and not on past states.

- Conditional independence of observation parameters:
\[
P(y_t|x_t, y_1, y_2, \ldots, y_{t-1}, x_1, x_2, \ldots, x_{t-1}) = P(y_t|x_t)
\]
(6)
Meaning that the observable variable \( y_t \) at time \( t \) depends only on the current hidden state \( x_t \). In other words, the probability of observing \( y \) while in hidden state \( x \) is independent of all other observable variables and past states.

Given the definition of HMM, the following issues arise:

**Evaluation problem:**

Given the HMM \( M = (A, B, \pi) \) and the observation sequence \( (y_1 \ldots y_t) \), how can we calculate the probability that Model \( M \) has generated sequence \( Y \)?

**Decoding problem:**

Given the HMM \( M = (A, B, \pi) \) and the observation sequence \( (y_1 \ldots y_t) \), how can we calculate the most likely sequence of hidden states \( X \) that produced this observation sequence \( Y \)?

**Learning problem:**

Given some training observation sequences and general structure of HMM (numbers of hidden and visible states), determine HMM parameters \( M = (A, B, \pi) \) that best fit training data.

The aforementioned problems can be solved with the following approaches:

**Evaluation problem:**

Given the HMM \( M = (A, B, \pi) \) and the observation sequence \( O = o_1 \ldots o_k \) trying to find the probability that Model \( M \) has generated sequence \( O \) by means of considering all hidden state sequences is impractical since the number of these sequences increases exponentially. For efficiently calculating this probability, the Forward-Backward HMM algorithm is used \([\text{Rabiner}, 1989]\): define the forward variable \( a_k(l) \) as the joint probability of the partial observation sequence \( O = o_1 \ldots o_k \) and that the hidden state at time \( k \) is \( S_l \),
\[
a_k(l) = P(o_1 \ldots o_k, q_k = S_l|S_l, a_k(l) = P(o_1, o_2 \ldots o_k, q_k = S_l)(6).
\]
Decoding problem:

Dealing with the decoding problem, we want to find the state sequence \( Q = q_1 q_2 \ldots q_k \) which maximizes \( P(Q | o_1 o_2 \ldots o_k) \), or equivalently \( P(Q, o_1 o_2 \ldots o_k) \). Instead of adopting the Brute force consideration of all paths, which requires exponential time, we can use the Viterbi algorithm [Viterbi, 1967]: define variable \( \delta_k(i) \) as the maximum probability of producing observation sequence \( O = o_1 o_2 \ldots o_k \) when moving along any hidden state sequence \( q_1 \ldots q_{k-1} \) and getting into \( q_k = S_i \).

\[
\delta_k(i) = \max \ P(q_1 \ldots q_{k-1}, q_k = S_i) \quad (7)
\]

Where max is taken over all possible paths \( q_1 \ldots q_{k-1} \).

Learning problem:

In the raining problem, given some training observation sequences \( O = o_1 o_2 \ldots o_k \) and general structure of HMM (numbers of hidden and visible states), we want to determine HMM parameters \( \lambda \), that best fit training data, that is maximizes \( P(O | \lambda) \). Since there is no algorithm for producing optimal parameter values, we use Baum-Welch algorithm [Rabiner, 1989], an iterative expectation-maximization algorithm, in order to find local maximum of \( P(O | \lambda) \).

3.2 NGI Module 2: The module for the insertion of new gestures

Besides using the set of predefined gestures supported by the module described in Section 3.1 (Figure 8), the game developers may wish to include extra gestures, more suitable for the games that they will design. To address this issue, we have designed NGI Module 2, which will offer developers the tools to record and insert their desired actions/gestures. The gestures will be performed in front of a Kinect 2 using a Capturing Module (this module will be described in later section) that will save the performed actions/gestures in the desired Xbox Event File (XEF). This procedure is described in Figure 11.

NGI Module for the insertion of new gestures

![Block diagram of NGI module 2](https://msdn.microsoft.com/en-us/library/dn785522.aspx)

In the lab experiments, we used the AdaBoostTrigger\(^2\) discrete method mentioned above and the

---

task was the recognition of 14 dynamic gestures: hand moving left, right, extent both arms, lean body left, right, etc.

**Recognition Task 1**

- Recognition of 14 dynamic gestures: hand moving left, right, extent both arms, lean body left, right, squat, jump, fly, throw ball, kick ball, tennis forehand, backhand, surfing and volley pass.
- During training time it accepts input tags, Boolean values.

We achieved accurate real-time recognition on the above gesture data set (~89.4%) and this fact proves methods’ potential for similar action/gesture recognition tasks.

### 3.3 Gesture Database

Complementary to the recognition module described in Section 3.1, we created a dataset with Microsoft Kinect recording of people performing 14 actions (directional, exergame gesture etc.), which are commonly encountered in a typical game (Table 1). We define an additional category, “neutral” category, to classify all frames where there is no motion indicating any of the distinct remaining action classes. Dataset contains 280 videos from 20 subjects. Each video begins with a close to neutral expression and proceeds to a peak expression. The total duration of its video is 5 seconds. Subjects were shown a short video with the aforementioned movements and afterwards they were asked to perform each movement according to their personal style, 3 times, in front of a Kinect sensor. Samples of the database showcasing game-related actions being captured by a Kinect for Xbox One device are shown in Figures 12-15.

<table>
<thead>
<tr>
<th>Gesture Category</th>
<th>Gestures</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directional</strong></td>
<td>Arm left, right, forward</td>
</tr>
<tr>
<td><strong>Virtual object avoidance</strong></td>
<td>Jump, Squat, Fly, Lean left, Lean right</td>
</tr>
<tr>
<td><strong>Virtual object manipulation</strong></td>
<td>Throw ball</td>
</tr>
<tr>
<td><strong>Exergame</strong></td>
<td>Tennis forehand, Tennis backhand, Volley pass, Kick ball, Surfing</td>
</tr>
</tbody>
</table>

*Table 1: NGI ProsocialLearn gesture database*
Figure 12. Direction gestures (forward, right, left)
Figure 13. Fly, jump, squat and lean gestures to avoid virtual obstacles
Figure 14. Throw gesture to manipulate virtual objects
3.4 Gesture recording platform

In the following paragraph, we will briefly explain how to setup Kinect Studio and record Kinect gesture data using Kinect SDK. It is worth to mention that .xef files generated by Kinect Studio are generally very large, a 10 second clip require more than 1GB even with the basic configuration for recording.

First, we have to open Kinect Studio v2 application (Figure 18) from windows menu. Then select ‘RECORD’ tab at the top if not already selected. We need to make sure that Kinect sensor is available and connected by clicking to ‘Connect to service’ button, as shown in Figure 16. Before start the recording, we have to select all the streams that need to be captured. Our NGI module uses only Depth, Body Index and Body frames, thus we can disable the recording of other feeds to get a smaller file size. Figure 17 illustrates the required streams. When everything is ready we can start the capturing process by clicking ‘record’ button. We need to make sure that each gesture clip has a unique name.

---

Figure 17. Kinect Studio Stream selection menu

Figure 18. Record, pause, stop and Playback.
3.5 Visual Gesture Builder

Visual Gesture Builder\(^5\) (VGB) generates data that applications use to perform gesture detection at run time. Even for simple cases, gesture detection is a challenging task that may require many lines of code to obtain reliable results, considering all of the different users and spaces that an application might encounter. By using a data-driven model, VGB shifts the emphasis from writing code to building gesture detection that is testable, repeatable, configurable, and database-driven. This method provides better gesture recognition and reduces development time.

VGB uses a number of detection technologies. The user selects the detection technologies to use—namely AdaBoostTrigger or RFRProgress—and tags frames in a clip related to a meaningful gesture, such as a jump or a kick. At the end of the tagging process, VGB builds a gesture database; with this database, an application can process body input from a user to, for example, detect a hit or swing progress.

3.5.1 AdaBoostTrigger

The AdaBoostTrigger is a detection technology that produces a binary or discrete result. It uses an Adaptive Boosting (AdaBoost) machine learning algorithm to determine when a user performs a certain gesture. During training time it accepts input tags, Boolean values, which mark the occurrence of a gesture, such as a hit. This marking or tagging is used to evaluate whether or not a gesture has happened and determines the confidence value of the event.

3.5.2 RFRProgress

The RFRProgress is a detection technology that produces an analog or continuous result. It uses the Random Forest Regression (RFR) machine learning algorithm to determine the progress of a gesture performed by a user.

3.5.3 Using Visual Gesture Builder

As described in previous section, we are able to record new gestures using Kinect Studio application. VGB uses skeleton data so we need to ensure that Body Frame and Body index streams are selected. Once we have recorded some clips including new gestures, we can import them into a VGB solution (Figure 19).

A solution contains a group of gesture projects (Figure 20). We are able to add new projects; one for each gesture we want to detect. In addition, it is a common practice to split data into a training set and a test set. The main choice we will need to make is whether we want to create a discrete or continuous gesture. A discrete gesture is Boolean in nature in that it is either happening or not with an associated confidence value whereas a continuous gesture provides a progress value and allows you to track progress optionally through multiple discrete gestures.
Once we have added a project with the wizard we can add a clip by right-clicking on the project and from there you can add clips to the project with a right click as illustrated in Figure 21.
Figure 21. Add clip to gesture

With a clip added we need to tag the clip to mark at which points the gesture is active (Figure 22). We can move the timeline over the clip and mark sections using keyboard shortcuts which are displayed on the app. Using this technique the more data you have and the better the tagging of that data is the better the result will be. If there are certain behaviors that you don’t want to detect in your gesture you can ‘train’ those out by not tagging them so the algorithm can learn some negative examples as well as positive.
When you have tagged your clips you can build the gesture database by either building the solution or a single project (Figure 23). Building VGB spits out some interesting information into its output window. This information includes details of features considered by the algorithm including a top ten features list. The result of the build is a database (with a .gbd file extension) which can be imported into an app using the Kinect SDK and used to detect your gesture.
In order to test your gesture without writing the code you can use the ‘Live Preview’ feature in VGB. This is located in the File menu as Live Preview. The resulted live feedback can be seen in Figure 24.

Figure 20. Building gesture dataset
Figure 21. Previewing a) arm left, b) fly and c) lean right gestures

With the graphs displaying progress for the continuous gesture and confidence for the discrete one. The following illustration (Figure 25) shows the context in which Visual Gesture Builder is used.

Figure 22. VGB process
4 Conclusions

In this report, a complete review of the designed NGI API was presented, with emphasis placed on the ways these modules could be used for ProsocialLearn games. Specifically, an overview of the developed NGI modules was presented, as well as a means of integration with a prototype prosocial game Path of Trust, providing useful insight for the future development. We have set a comprehensive and thorough example on how to design an NGI system, choose algorithms and hardware, record and add new gestures into dataset in a way that will allow future game developments to rapidly prototype new prosocial games that will use a variety of interaction styles. After setting this paradigm, we have developed, and delivered both a gesture recognition module (Figure 8) and database, which are readily available for adding simple gesture-driven interaction within future prosocial games, as well as a gesture recording module (Figure 11), for game developers wanting to incorporate new gestures not included in our released dataset. We believe the components to be sufficient for developers to support the use of natural user interaction in future games, as they can either use our pre-defined dataset for simple games, or have the recognition module accurately classify new gestures altogether, therefore expanding the possibilities for new and exciting gameplay options offered through the support of the Kinect sensor.

Finally, we have to note that the main objective of this deliverable is to present simple natural user interfaces that enable players to interact with the game. The identification of players affective state through gesture analysis or facial expression analysis was presented in deliverables D3.2 “1st Prosocial affect fusion and player modeling” and D3.2 “2nd Prosocial affect fusion and player modeling” of WP3.
5 References


