Realizarea de exerciții prin joc pentru sistemul TRAVEE

Context platforma TRAVEE

Sistemul TRAVEE este alcătuit din mai multe componente software și hardware. Pentru gestionarea centralizată a acestora, sistemul conține o platformă web ce oferă terapeuților accesul de control la parametrii exercițiilor prin joc.

Două dintre principalele componente ale sistemului sunt:

1. Efectuarea sesiunilor de recuperare neuromotorie cu exerciții prin joc în **context clinic**. Aici, pacienții sunt asistați la orice moment de timp de către terapeuți. Aceștia din urmă au posibilitatea ajustării valorilor pentru parametrii exercițiilor prin joc în timpul desfășurării lor. Un exemplu de utilizare în context clinic se poate observa în Figura 1.
2. Efectuarea sesiunilor de recuperare neuromotorie cu exerciții prin joc în **context de lucru de acasă**. În această situație, pacienții utilizează independent dispozitivul VR și sistemul pentru desfășurarea sesiunilor. Terapeuții realizează o prescripție de exerciții prin joc ce sunt pre-ajustate astfel încât să corespundă nevoilor pacienților.

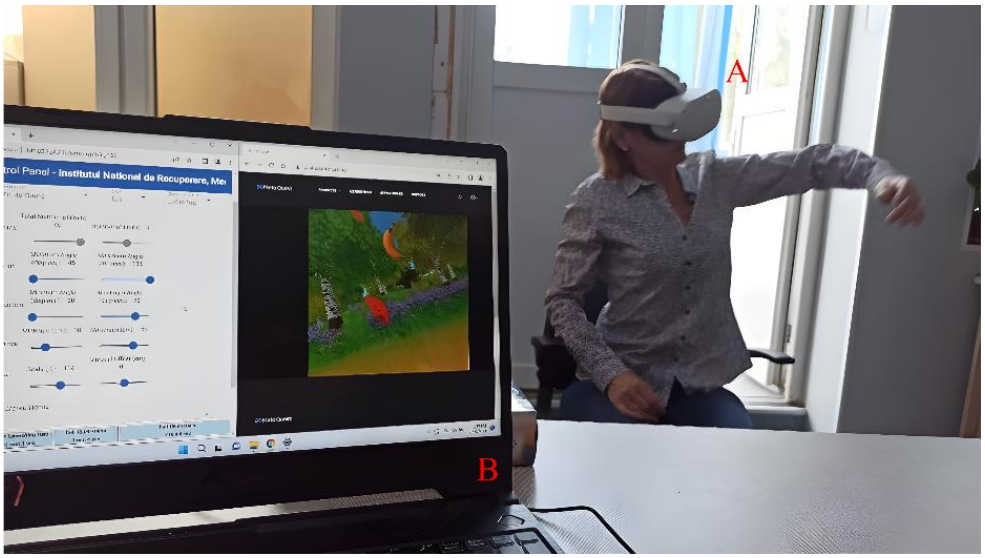

Figura 1. Marcajul A: Dispozitivul VR Meta Quest 2, utilizat de către un pacient în context clinic. Marcajul B: Interfața cu utilizatorul oferită de către platforma web de administrare a sistemului TRAVEE, executată pe un laptop operat de către terapeuți.

Utilizare context clinic

Interfața cu utilizatorul oferită de către platforma web de administrare în context clinic poate fi vizualizată în Figura 2.

Această interfață este specifică fiecărui exercițiu prin joc și este creată dinamic, astfel, dezvoltatorul exercițiului prin joc nu trebuie sa creeze o interfață HTML pentru exercițiu ci trebuie să creeze un document în format .json, care conține descrierea parametrilor exercițiului. Acest document în format .json este analizat în detaliu în capitolele următoare.

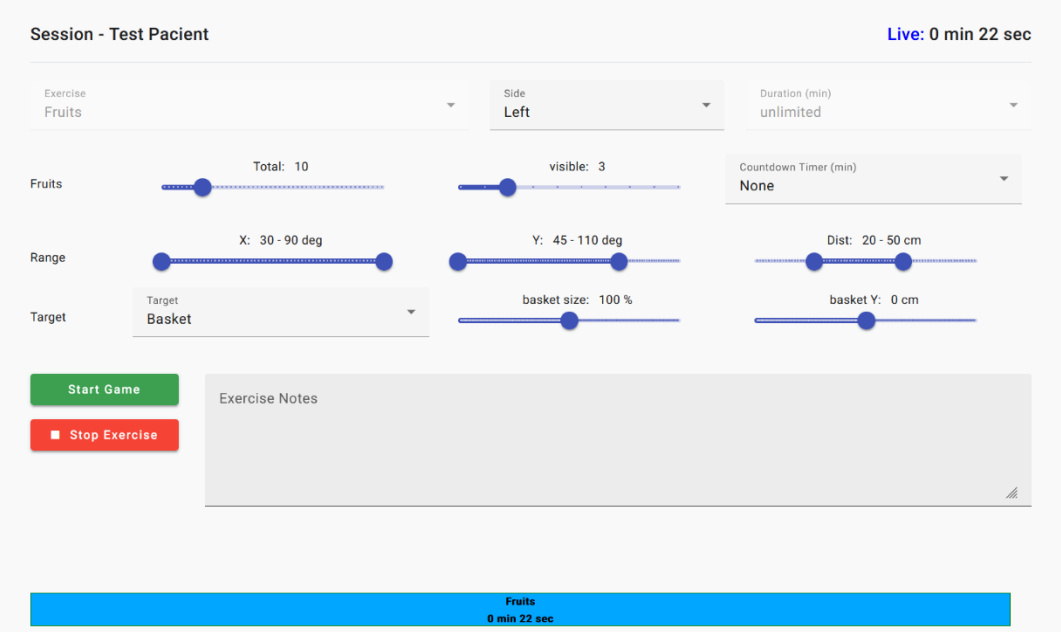

Figura 2. Interfața cu utilizatorul pentru exercițiul prin joc „Fructele”, în context clinic. Codul sursă al acestui joc este pus la dispoziție pentru voi în arhiva oferită de echipa TRAVEE.

La momentul desfășurării unei sesiuni de recuperare în context clinic, terapeutul alege din platforma web de administrare opțiunea de a începe un exercițiu și vizualizează interfața cu utilizatorul ce poate fi consultată în Figura 2. Terapeutul poate stabili valorile parametrilor specifici exercițiului prin joc și alege opțiune de începere a unei sesiuni de joc cu exercițiul selectat prin apăsarea butonului ce conține textul „Start Game”.

După alegerea acestei opțiuni, interfața se schimba. Aceasta poate fi vizualizată în Figura 3. Butonul ce conține textul „Start Game” dispare și apare un buton ce conține textul „Stop Game”. Aceste 2 butoane sunt specifice fiecărui exercițiu prin joc și se pot crea dinamic pe baza descrierii din documentul in format .json, ce este descris în capitolele următoare.

De asemenea, în interfața cu utilizatorul ce apare după începerea unei sesiuni de joc, terapeutul nu mai are acces la modificarea tuturor parametrilor, prin comparație cu pasul anterior. La momentul desfășurării unei sesiuni de joc, terapeutul poate avea acces doar la parametri a căror modificare nu necesită începerea sesiunii. În general, este de dorit să existe posibilitatea modificării tuturor parametrilor la momentul desfășurării unui exercițiu, dar nu întotdeauna este posibil tehnic acest lucru.

De asemenea, în secțiunea „Exercise Notes” din interfață, apare o notiță completată automat de către sistem, care înregistrează câteva informații despre acțiunea realizată de către terapeut. După apăsarea butonului „Stop Game”, interfața se schimbă după cum se poate observa în Figura 4.

Figura 3. Interfața cu utilizatorul pentru exercițiul prin joc „Fructele”, în context clinic, la momentul desfășurării unei sesiuni de joc.

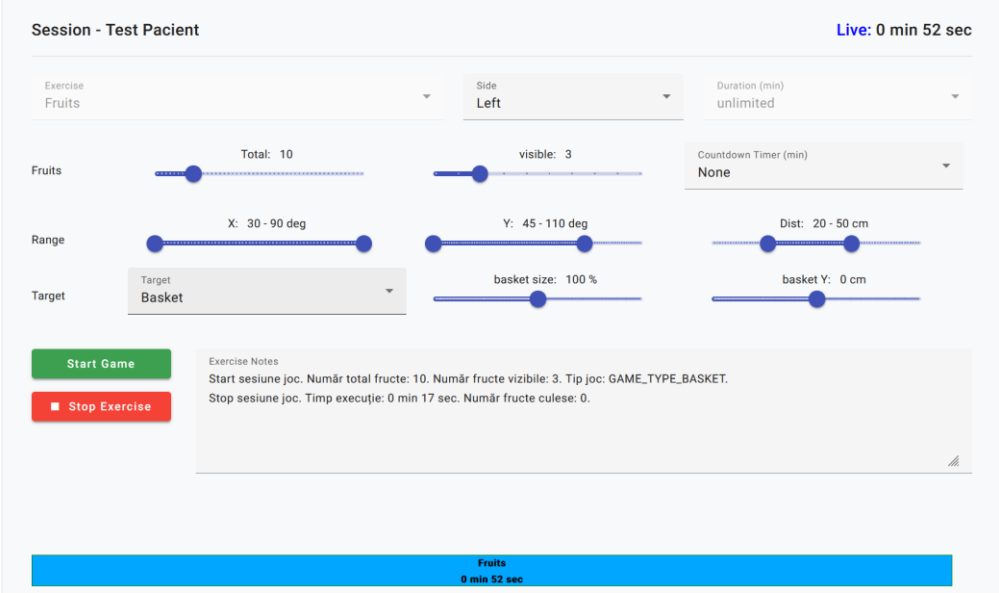

Figura 4. Interfața cu utilizatorul pentru exercițiul prin joc „Fructele”, în context clinic, după închiderea unei sesiuni de joc.

Accesul la această platformă, în scopul dezvoltării exercițiilor prin joc necesită realizarea unui set de pași specifici, ce îngreunează procesul de dezvoltare în sine. De asemenea, procesul de testare devine încet, deoarece necesită conexiunea cu un server, accesul la platforma web de administrare și accesul la evenimentele înregistrate de către sistem pentru procesul de depanare. Pentru a ușura dezvoltarea, echipa TRAVEE vă pune la dispoziție un sistem incorporat direct în editorul Unity, ce simulează mare parte din interfața cu utilizatorul a platformei web de administrare. Acest sistem de dezvoltare este descris în capitolul următor.

Utilizare context lucru de acasă

În procesul de lucru de acasă, terapeutul stabilește o prescripție, ce cuprinde un set de jocuri, împreună cu valorile parametrilor de configurare pentru acestea. Interfața de gestionare a unei prescripții este similară cu cea utilizată pentru stabilirea parametrilor în context clinic și permite modificarea tuturor valorilor acestor parametri.

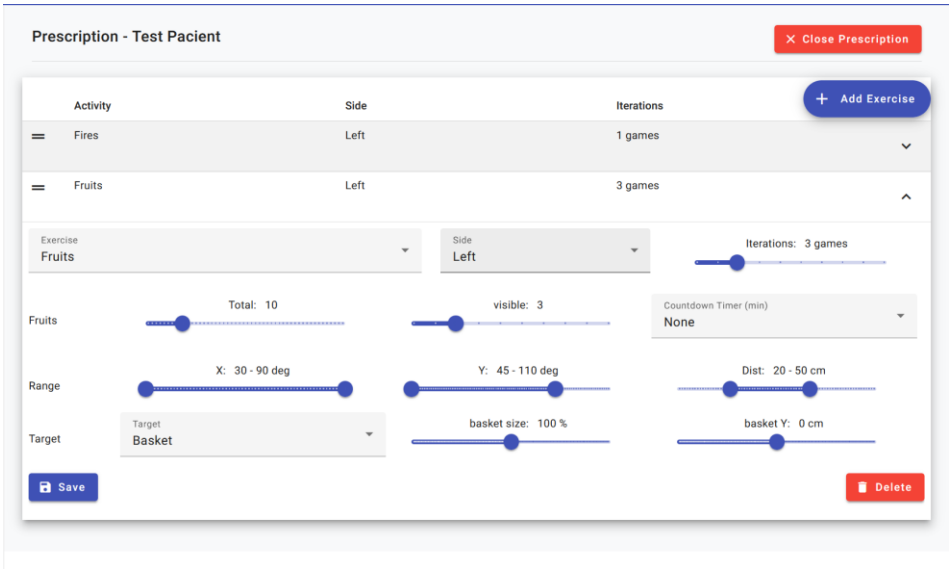

Figura 5. Interfața cu utilizatorul pentru realizarea unei prescripții pentru exercițiul prin joc „Fructele”, în context de lucru de acasă.

Similar precum interfața destinată utilizării sistemului în context clinic, această interfață este realizată dinamic, pe baza documentului în format .json, ce este analizat în capitolele următoare.

Dezvoltarea unui exercițiu prin joc

Dispozitiv hardware

Sistemul TRAVEE este proiectat special pentru dispozitivele VR de tip Meta Quest 2 și Meta Quest 3. Acestea conțin o unitate centrală de procesare și o unitate de procesare grafică, astfel că utilizarea lor este complet autonomă și nu necesită legarea dispozitivelor la un procesor extern.

Suplimentar, dispozitivele VR de tip Meta Quest 2 și 3 incorporează un sistem de recunoaștere a mișcării mâinilor, ce cuprinde recunoașterea mișcării degetelor, palmei și încheieturii. Acest sistem este utilizat de sistemul TRAVEE pentru desfășurarea sesiunilor de recuperare cu exerciții prin joc.

Astfel, pentru dezvoltarea și testarea exercițiilor prin joc realizate, este necesar accesul fizic la un dispozitiv hardware de tip Meta Quest.

Inițializare motor Unity pentru dezvoltare

Folosiți uneltele de dezvoltare descrise în continuare:

1. Utilizați motorul de joc *Unity* cu versiunea *2022.3.Xf1* LTS.
2. Creați un nou proiect de tip *3D Built-In Render Pipeline*.
3. Instalați pachetul *Meta XR All-in-One SDK*, versiunea *74.0.0*.

4. Stabiliți setările de configurare ale aplicației pentru sistemul Oculus. Aici există un tutorial foarte detaliat pe acest subiect.
5. Copiați conținutul arhivei pusă la dispoziție de echipa TRAVEE în proiectul vostru.

Platforma TRAVEE

Platforma TRAVEE este compusă dintr-un set de sisteme ce intercomunică la orice moment de timp. Aplicațiile de tip exercițiu prin joc au la dispoziție o interfață abstractă pentru a facilita dezvoltarea acestora. Descrierea structurii în care astfel de aplicații comunică cu celelalte componente ale sistemului este prezentată în Figura 6.

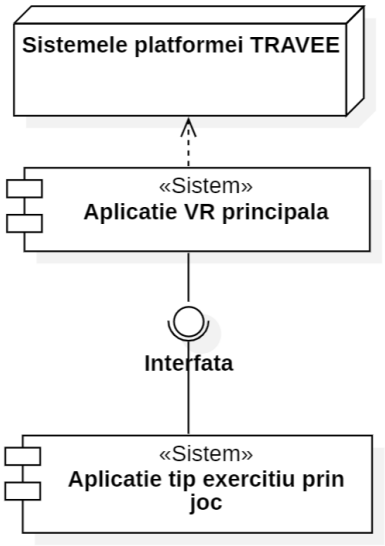

Figura 6. Structura componentelor cu care comunica aplicația de tip exercițiu prin joc

Structura

Aplicațiile de tip exercițiu prin joc sunt dezvoltate sub forma unor module ce trebuie să se regăsească în două componente distincte ale platformei TRAVEE:

1. În aplicația VR, prin introducerea codului sursă al aplicației de tip exercițiu prin joc în codul sursă al aplicației VR principale;
2. În platforma web de administrare, în care trebuie introduse informații de descriere a aplicației, de exemplu: numele jocului, numele scenei principale a jocului. Aceste informații sunt descrise într-o secțiune următoare.

Aplicația VR

Aplicațiile de tip exercițiu prin joc trebuie dezvoltate pentru motorul de jocuri Unity și destinate modelului de cască VR Meta Quest.

Pentru comunicarea cu aplicația VR principală, vizibilă în Figura 6, dezvoltatorul trebuie să aibă stabilită o scenă principală de pornire a aplicației, care să conțină un obiect cu numele *SessionContainer*, așa cum se poate vedea în Figura 7. În situația în care scena se schimbă pe perioada execuției aplicației, acest obiect trebuie să nu fie șters. Atașat acestui obiect, trebuie să se

afle o componentă care poate avea orice nume, dar care trebuie să aibă structura prezentată în Listarea 1. Această structură se poate găsi și în fișierul *SessionContainer.cs*. Atașarea componentei asupra obiectului din scenă cu numele *SessionContainer* se poate observa în Figura 7.

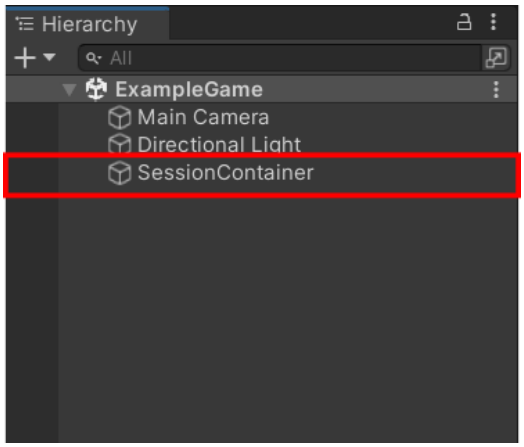

Figura 7. Un exemplu de ierarhie de componente ce conține un obiect cu numele *SessionContainer*

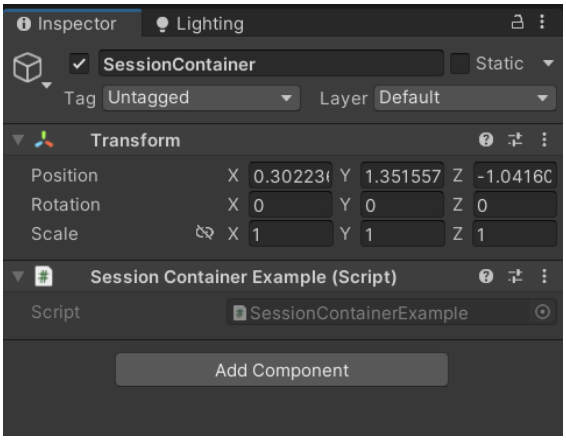

Figura 8. Fereastra de inspecție a componentelor atașate obiectului *SessionContainer*. Se poate observa că acest obiect are atașată o componentă cu numele *SessionContainerExample*, ce are structura prezentată în Listarea 1

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using UnityEngine;

namespace ExampleGame
{
    /// <summary>
    /// An example of input data.
    /// Only int, bool and list data types are allowed at this time.
    /// </summary>
    public class InputData
    {
        /// <summary>
        /// Example int.
        /// </summary>
        public int ExampleInt;
```

```
        /// <summary>
        /// Example bool.
        /// </summary>
        public int ExampleBool;

        /// <summary>
        /// Example list.
        /// This attribute only stores the index of the selected option from the
list.
        /// </summary>
        public int ExampleList;

        /// <summary>
        /// Default attribute. It is optional.
        /// </summary>
        public int duration;

        /// <summary>
        /// Default attribute. Represents a list of options for selecting the hand
        /// with which the exercise is performed.
        /// </summary>
        public BodySide BodySide;
    }

    /// <summary>
    /// An example of output data.
    /// Only int and bool data types are allowed at this time.
    /// </summary>
    [Serializable]
    public class GameSessionOutputData : GameSessionOutputDataBase
    {
        public int ExampleInt;
        public bool ExampleBool;
        public List<string> ExampleList;
    }

    /// <summary>
    /// An example of output data.
    /// </summary>
    public class OutputData
    {
        /// <summary>
        /// Example int.
        /// </summary>
        public int ExampleInt;
    }

    public class SessionContainer : SessionContainerBase
    {
        /// <summary>
        /// Object that contains the input parameter values for the game.
        /// </summary>
        [SerializeField]
        protected InputData _inputData;

        /// <summary>
        /// Object assigned with controlling game behavior în a clinical context.
        /// </summary>
```

```
        [SerializeField]
        protected ClinicalUseController _clinicalUseController;

        /// <summary>
        /// Object assigned with controlling game behavior în a home use context.
        /// </summary>
        [SerializeField]
        protected HomeUseController _homeUseController;

        /// <summary>
        /// Internal object used to simplify the management of the context
        /// for which the game is intended.
        /// </summary>
        protected GameManagerBase _mainController;

        /// <summary>
        /// Event called when the main VR application notifies the exergame to
start.
        /// </summary>
        /// <param name="json">The input data of the exergame în JSON
format.</param>
        public override void OnInit(string json)
        {
            _inputData = JsonUtility.FromJson<InputData>(json);

            _clinicalUseController.Init(_inputData);

            _mainController = _clinicalUseController;
        }

        /// <summary>
        /// Event called when the main VR application notifies the exergame to
start
        /// în a home use context.
        /// </summary>
        /// <param name="json">The input data of the exergame în JSON
format.</param>
        public override void OnInitHomeUse(string json)
        {
            _inputData = JsonUtility.FromJson<InputData>(json);

            _homeUseController.Init(_inputData);

            _mainController = _homeUseController;
        }

        /// <summary>
        /// Event called when the main VR application notifies the exergame to
update.
        /// </summary>
        /// <param name="json">The input data of the exergame în JSON
format.</param>
        public override void OnUpdate(string json)
        {
            _inputData = JsonUtility.FromJson<InputData>(json);

            _clinicalUseController.UpdateGame(_inputData);
        }
```

```
        /// <summary>
        /// Event called when the main VR application notifies the exergame to
pause.
        /// </summary>
        public override void OnPause(bool pause)
        {
            // Optional
            // Game logic
        }

        /// <summary>
        /// Event called when the main VR application notifies the
        /// exergame to start a custom event.
        /// </summary>
        /// <param name="eventName">The name of the custom event.</param>
        public override void OnCustomEvent(string eventName)
        {
            if (eventName == "startgame") {
                _clinicalUseController.StartGame();
            }

            if (eventName == "stopgame") {
                _clinicalUseController.StopGame();
            }
        }

        /// <summary>
        /// Event called when the main VR application notifies the exergame to
stop.
        /// </summary>
        public override void OnStop()
        {
            if (_onStop == null) {
                return;
            }

            var outputData = _mainController.OutputData;

            string output = JsonConvert.SerializeObject(outputData);

            _onStop(output);
        }
    }
}
```

Listarea 1. Codul sursă al componentei *SessionContainerExample*

Codul sursă al componentei SessionContainer trebuie să respecte structura descrisă în Listarea 1. Este recomandat ca această clasă să moștenească clasa *SessionContainerBase*, ce este atașată în fișierul *SessionContainerBase.cs*.

Este absolut necesar să introduceți codul componentei *SessionContainer* într-un namespace dedicat exercițiului vostru prin joc. De fapt, tot codul specific exercițiului dezvoltat trebuie introdus în acest namespace.

Structura menționată, pe care trebuie să o respecte clasa, se referă la faptul că aceasta trebuie să conțină obligatoriu 5 metode, ce au rolul să fie apelate prin metoda *SendMessage()* [1] de către sistemul de comunicare, implementat în aplicația VR principală. Cele 5 metode sunt descrise în detaliu în Tabelul 1.

Context	Nume	Descriere
Clinic	`void OnInit(string json)`	O componentă din aplicația VR principală are rolul de a schimba scena principală de pornire a jocului realizat de către dezvoltator. După ce această scenă este încărcată cu succes, începerea jocului se realizează doar după apelarea metodei *OnInit()* din obiectul cu numele *SessionContainer*.
	`void OnCustomEvent(string eventName)`	După apelarea metodei *OnInit()*, aplicația VR principală poate apela metoda *OnCustomEvent()* pentru gestionarea evenimentelor specifice jocului. Din această categorie fac parte și evenimentele „Start Game” și „Stop Game”, create prin apăsarea butoanelor specifice din interfața platformei web de administrare de către terapeuți.
	`void OnUpdate(string json)`	În situația în care se modifică valorile parametrilor de configurare pe perioada execuției, se apelează metoda *OnUpdate()*. Este important de menționat că dezvoltatorul poate alege ce parametri sa poată fi modificați în timpul execuției aplicației.
	`void OnStop()`	In momentul în care terapeutul alege încheierea exercițiului, prin apăsarea unui buton de „Stop” din interfața web, se apelează metoda *OnStop()*.
Acasă	`void OnInitHomeUse(string json)`	Analog cu metoda *OnInit()*, descrisă mai sus în acest tabel. Apelarea acestei metode se realizează doar când sistemul TRAVEE este utilizat în context de lucru de acasă.

Tabelul 1. Descrierea metodelor ce trebuie implementate în vederea comunicării cu sistemul TRAVEE

Vă rugăm să utilizați structura de cod din arhiva pusă la dispoziție de echipa TRAVEE, mai exact, scheletul componentelor *ClinicalUseController* și *HomeUseController*. Aceste clase sunt documentate în cod. Vă rugăm să consultați codul pus la dispoziție pentru jocul „Fructele” pentru a avea un exemplu de proiectare și de pornire în dezvoltarea exercițiului prin joc realizat de voi.

Platforma web de administrare

În platforma web de administrare, trebuie introduse un set de informații. Acestea vor fi introduse de către echipa TRAVEE ulterior în platformă. În acest moment, aceste informații pot fi transmise într-un fișier text. Informațiile sunt:

1. Numele aplicației de tip exercițiu prin joc;
2. Numele scenei principale în care se deschide aplicația;
3. Un set de date ce descriu structura datelor de intrare specifice aplicației de tip exercițiu prin joc. Datele sunt transmise în format JSON. Un exemplu de astfel de date este descris în Listarea 2.

```
{
    "inputParameters": [
        {
            "name": "BodySide",
            "text": [
                {
                    "language": "en",
                    "value": "Side"
                },
                {
                    "language": "ro",
                    "value": "Mână"
                }
            ],
            "type": "list",
            "value": [
                {
                    "text": [
                        {
                            "language": "en",
                            "value": "Left"
                        },
                        {
                            "language": "ro",
                            "value": "Stânga"
                        }
                    ],
                    "value": "0"
                },
                {
                    "text": [
                        {
                            "language": "en",
                            "value": "Right"
                        },
                        {
                            "language": "ro",
                            "value": "Dreapta"
                        }
                    ],
                    "value": "1"
                }
            ],
            "defaultValue": "0",
            "isImplicit": true,
            "runtimeUpdate": true
        },
        {
            "name": "IterationCount",
            "text": [
                {
                    "language": "en",
                    "value": "Iterations"
                },
                {
                    "language": "ro",
                    "value": "Iterații"
                }
```

```
            ],
            "finalText": [
                {
                    "language": "en",
                    "value": "games"
                },
                {
                    "language": "ro",
                    "value": "jocuri"
                }
            ],
            "type": "int",
            "defaultValue": "1",
            "minValue": "1",
            "maxValue": "200",
            "isImplicit": true,
            "isHomeUseOnly": true
        },
        {
            "name": "ExampleInt",
            "text": "Example int",
            "type": "int",
            "defaultValue": "1",
            "minValue": "1",
            "maxValue": "5",
            "runtimeUpdate": true,
            "categoryName": "ExampleCategory1",
            "observedEvents": [
                {
                    "name": "show",
                    "activeEventNames": [
                        "onOption0Selected"
                    ]
                }
            ]
        },
        {
            "name": "ExampleBool",
            "text": "Example Bool",
            "type": "bool",
            "defaultValue": false,
            "categoryName": "ExampleCategory1",
            "observedEvents": [
                {
                    "name": "enable",
                    "activeEventNames": [
                        "onNoGameIsPlaying"
                    ]
                }
            ]
        },
        {
            "name": "ExampleList",
            "text": "Example List",
            "type": "list",
            "value": [
                {
                    "text": "Option 0",
```

```
                    "value": "0"
                },
                {
                    "text": "Option 1",
                    "value": "1"
                },
                {
                    "text": "Option 2",
                    "value": "2"
                }
            ],
            "defaultValue": "0",
            "categoryName": "ExampleCategory2",
            "emittedEvents": [
                {
                    "value": "0",
                    "eventNames": [
                        "onOption0Selected"
                    ]
                }
            ]
        }
    ],
    "inputParameterCategories": [
        {
            "name": "ExampleCategory1",
            "text": [
                {
                    "language": "en",
                    "value": "Example Category 1"
                },
                {
                    "language": "ro",
                    "value": "Exemplu categorie 1"
                }
            ]
        },
        {
            "name": "ExampleCategory2",
            "text": [
                {
                    "language": "en",
                    "value": "Example Category 2"
                },
                {
                    "language": "ro",
                    "value": "Exemplu categorie 2"
                }
            ]
        }
    ],
    "buttons": [
        {
            "name": "startgame",
            "text": "Start Game",
            "type": "start",
            "observedEvents": [
                {
```

```
                    "name": "show",
                    "activeEventNames": [
                        "onNoGameIsPlaying"
                    ]
                }
            ]
        },
        {
            "name": "stopgame",
            "text": "Stop Game",
            "type": "stop",
            "observedEvents": [
                {
                    "name": "show",
                    "activeEventNames": [
                        "onGameIsPlaying"
                    ]
                }
            ]
        }
    ]
}
```

Listarea 2. Descrierea în format .json a unui set de date de intrare

Descrierea informațiilor specifice parametrilor de configurare ai exercițiului prin joc se regăsește în Tabelul 2.

Nume	**Opționalitate**	**Descriere**
„name”	Obligatoriu	Numele parametrului utilizat pentru referirea lui în aplicația VR. Este utilizat pentru comunicarea valorii dintre aplicația web și cea VR.
„text”	Obligatoriu	Descrierea text a parametrului ce apare în aplicația web pentru terapeut. Este utilizat pentru a descrie în limbaj natural atributul astfel încât să poată fi înțeles ușor de către terapeut. *Acest atribut poate fi tradus în mai multe limbi prin sistemul de traducere descris mai jos.*
„type”	Obligatoriu	Tipul de date al parametrului. Sunt posibile 4 tipuri de date: „int”, „ivec2”, „bool” și „list”: • Tipul de date „int” permite valori numerice; • Tipul de date „ivec2” permite o pereche de valori numerice; • Tipul de date „bool” permite o valoare logica de adevăr. Mai exact, permite doar valorile „adevărat” sau „fals”; • Tipul de date „list” permite o listă de șiruri de caractere.
„finalText”	Obligatoriu	Este o descriere text suplimentară ce este afișată în aplicația web după afișarea valorii curente a parametrului. Un exemplu de utilizare este afișarea unității de măsură pentru valoarea parametrului: *distanta: 100 „cm”* Șirul de caractere „cm” din exemplul de mai sus este specificat prin atributul „finalText”. *Acest atribut poate fi tradus în mai multe limbi prin sistemul de traducere descris mai jos.*

„value”	Obligatoriu pentru tipul de date „list”	Acest atribut este specific doar tipului de data „list” și descrie opțiunile pe care le poate avea parametrul sub forma unor structuri de date în felul următor: • "value": [{ "text": "Left", "value": "0" }, { "text": "Right", "value": "1" }] • "value": [{ "text": "Any", "value": "0" }, { "text": "Pumn deschis", "value": "1" }, { "text": "Pumn închis", "value": "1" }]
„defaultValue”	Opțional, valabil doar pentru tipurile de date „int”, „bool” și „list”	Reprezintă valoarea implicită pentru parametru. Este utilizat pentru afișarea implicită a valorii în aplicația web. Depinde de tipul de date: • Pentru tipul de date „int”, valoarea trebuie să fie numerică; • Pentru tipul de date „bool”, valoarea trebuie să fie „true” sau „false”; • Pentru tipul de date „list”, valoarea trebuie să fie un indice în listă, ce pornește de la valoarea 0. Pentru tipul de date „ivec2”, acest atribut este înlocuit de alte două atribute: „minDefaultValue” și „maxDefaultValue” descrise mai jos.
„minValue”	Opțional, valabil doar pentru tipul de date „int”, obligatoriu pentru tipul de date „ivec2”	Specifică valoarea minimă pe care o poate avea parametrul. Este utilizat pentru limitarea valorii pe care o poate avea parametrul în aplicația web. În situația în care este specificată perechea „minValue” și „maxValue”, în aplicația web este afișat un control de tip „slider”.
„maxValue”	Opțional, valabil doar pentru tipul de date „int”, obligatoriu pentru tipul de date „ivec2”	Specifică valoarea maximă pe care o poate avea parametrul. Este utilizat pentru limitarea valorii pe care o poate avea parametrul în aplicația web. În situația în care este specificată perechea „minValue” și „maxValue”, în aplicația web este afișat un control de tip „slider”.
„minName”, „maxName”	Obligatoriu pentru tipul de date „ivec2”	Reprezintă numele utilizat de către aplicația VR pentru a referi perechea de valori a parametrului. Pentru tipul de date „ivec2” se utilizează un nume pentru fiecare valoare, conform acestor două atribute.
„minDefaultValue”, „maxDefaultValue”	Opțional pentru tipul de date „ivec2”	Similar cu atributul „defaultValue”, reprezintă valoarea implicită pentru perechea de valori a unui atribut ce are tipul de data „ivec2”.
„runtimeUpdate”	Opțional	Specifică faptul că acest atribut poate fi modificat în timpul execuției exercițiului. Este utilizat pentru a permite din interfața aplicației web să se modifice valoarea parametrului după pornirea exercițiului. Implicit, se consideră că un atribut nu poate fi modificat în timpul execuției.
„categoryName”	Opțional	Specifică o categorie din care face parte parametrul. Este utilizat pentru a grupa parametrii pe categorii în interfața aplicației web utilizată de către terapeut. Implicit, se consideră ca toți parametrii fac parte din categoria „General”.

		Categoriile sunt definite într-o secțiune specială din fișierul .json. Aceasta este descrisă mai jos.
„isImplicit”	Opțional	Specifică o categorie specială și este utilizat pentru a afișa parametrul într-o zonă specială din interfață. Implicit, se consideră că parametrii nu fac parte din această categorie specială.
„requiresHDC”	Opțional	Specifică faptul că pentru acest atribut este necesară utilizarea aplicației HDC. Implicit, parametrii nu necesita utilizarea acestei aplicații.
„hdcId”	Opțional	Specifică dispozitivul hardware necesar pentru parametru printr-un identificator. Opțiunile posibile sunt: • EEG; • IMU; • Kinect.
„homeUseOnly”	Opțional	Specifică faptul că acest atribut apare doar în interfața cu utilizatorul pentru gestionarea prescripției exercițiului prin joc în contextul de lucru pentru acasă. Implicit, acest parametru este false. In situația in care atributul este true, parametrul nu apare în interfața platformei web de administrare la momentul desfășurării unei sesiuni de joc în context clinic.

Tabelul 2. Descrierea informațiilor specifice datelor de intrare

Descrierea informațiilor specifice evenimentelor se regăsește în Tabelul 3.

Nume	**Opționalitate**	**Descriere**
„name”	Obligatoriu	Numele evenimentului utilizat pentru referirea lui în aplicația VR, la apăsarea lui în platforma de administrare. Este utilizat pentru comunicarea valorii dintre aplicația web și cea VR.
„text”	Obligatoriu	Descrierea text a butonului ce apare în aplicația web pentru terapeut. Este utilizat pentru a descrie în limbaj natural butonul astfel încât să poată fi înțeles ușor de către terapeut. *Acest atribut poate fi tradus în mai multe limbi prin sistemul de traducere descris mai jos.*
„type”	Obligatoriu	Tipul butonului, ce stabilește culoarea lui. Această opțiune are scop strict decorativ. Sunt posibile 2 tipuri de buton: „start” și „stop”: • Tipul de buton „start” are culoarea verde; • Tipul de date „stop” are culoarea roșie.

Tabelul 3. Descrierea informațiilor specifice butoanelor

Atributele „observedEvents” și „emittedEvents” sunt specifice unui sistem de constrângeri dezvoltat în sistemul TRAVEE, ce este complex și depășește scopul acestei documentații. Pentru a nu îngreuna procesul de dezvoltare al exercițiului prin joc mai mult decât este în prezent, vă rugăm să utilizați cele 2 butoane de pornire și închidere a unei sesiuni de joc, așa cum sunt puse la dispoziție în documentul din Listarea 2.

Sistemul de constrângeri este complex și nu este complet implementat în editorul de Unity, astfel că toate situațiile care îl necesită, în situația în care acestea există, vor fi incorporate ulterior dezvoltării de către voi. În situația în care doriți să utilizați explicit acest sistem, vă rugăm să contactați un membru din echipa TRAVEE.

Sistemul de traducere a atributelor text

Sistemul TRAVEE permite traducerea tuturor atributelor text care sunt afișate în interfață. În Tabelul 2 și în Tabelul 3, la descrierea atributelor traductibile, a fost menționat explicit faptul că acele atribute suportă posibilitatea să fie traduse.

Traducerea unui atribut traductibil este complet opțională. Altfel spus, sistemul de traducere a atributelor text poate fi utilizat doar în situația în care se dorește traducerea unuia dintre atributele traductibile. În momentul în care nu se dorește o traducere explicită, se utilizează valoarea specificată atributului, indiferent de limba utilizată de către sistem.

În Listarea 3 se poate consulta descrierea a doi parametri de configurare. Parametrului cu numele „Untranslated” i se specifică o valoare text, care este utilizată indiferent de limba de traducere a sistemului TRAVEE. Pe de altă parte, parametrului cu numele „Translated” i se specifică 2 opțiuni de traducere, una pentru limba română și una pentru limba engleză.

```
{
    "inputParameters": [
        {
            "name": "Untranslated",
            "text": "Untranslated",
            "type": "int",
            "defaultValue": "1"
        },
        {
            "name": "Translated",
            "text": [
                {
                     "language": "en",
                     "value": "Translated"
                },
                {
                     "language": "ro",
                     "value": "Tradus"
                }
            ],
            "type": "int",
            "defaultValue": "1"
        }
    ]
}
```

Listarea 3. Descrierea în format .json a sistemului de traducere a atributelor text pentru 2 parametri de configurare

Pentru specificarea explicită a unei traduceri, se introduce un set de elemente care conțin atributele descrise în Tabelul 4.

Nume	Opționalitate	Descriere
„language”	Obligatoriu	Limba în care se traducere atributul. Opțiunile pot fi: • „en” pentru traducere în limba engleză; • „ro” pentru traducere în limba română. La momentul redactării acestei documentații, sistemul TRAVEE permite traducerea doar în cele 2 limbi menționate.

„value”	Obligatoriu	Descrierea text a traducerii în limba specificată prin atributul descris mai sus.

Tabelul 4. Descrierea atributelor specifice unei limbi utilizate în sistemul de traducere a atributelor text

Categoriile parametrilor de configurare

Din punct de vedere vizual, parametrii de configurare sunt grupați în categorii, astfel încât să fie ușor de urmărit de către terapeuții care utilizează sistemul TRAVEE. În Figura 2, se poate observa gruparea parametrilor de configurare ai exercițiului prin joc pe 3 categorii și în partea stângă a figurii se pot observa numele categoriilor: *„Fruits”*, *„Range”* și *„Target”*.

Un parametru de configurare face parte obligatoriu dintr-o singură categorie. Aceasta din urmă poate fi specificată explicit prin utilizarea atributului *„categoryName”*. În situația în care nu este precizată nicio categorie explicită prin utilizarea atributului, parametrul de configurare face parte implicit din categoria *„General”*.

Specificarea categoriei prin atributul „categoryName” se poate realiza în două variante. În prima variantă se poate specifica în atributul „categoryName” un nume de identificare unic care este utilizat pentru numele categoriei în care se regăsește parametrul de configurare. În situația în care mai mulți parametri folosesc același nume de identificare, toți aceștia sunt grupați automat. În momentul în care se dorește traducerea numelui de identificare în mai multe limbi, trebuie creată o secțiune specială numită *„inputParameterCategories”*, așa cum este prezentat în Listarea 2. Descrierea atributelor de configurare ale unei categorii din secțiunea specifică categoriilor este prezentată în Tabelul 5.

Nume	**Opționalitate**	**Descriere**
„name”	Obligatoriu	Numele de identificare al categoriei, utilizat pentru referirea lui în atributul „categoryName” din descrierea unui parametru de configurare.
„text”	Obligatoriu	Descrierea text a categoriei ce apare în aplicația web pentru terapeut. Este utilizat pentru a descrie în limbaj natural categoria astfel încât să poată fi înțeleasă ușor de către terapeut. *Acest atribut poate fi tradus în mai multe limbi prin sistemul de traducere descris mai jos.*

Tabelul 5. Descrierea atributelor specifice categoriilor in care sunt grupati parametrii de configurare

Înlocuirea interfeței web în procesul de dezvoltare

Pentru a facilita procesul de dezvoltare, este creată o componentă specială, disponibilă în fișierele *InputDataContainer.cs*, ce conține codul componentei și *InputDataEditor.cs*, ce conține codul pentru desenarea interfeței componentei în editorul din Unity. Această componenta permite controlul datelor de intrare pentru joc direct din interfața editorului de Unity, cu un comportament asemănător controlului interfeței web de administrare.

Pentru a utiliza această componentă, cele două fișiere menționate mai sus trebuie introduse în proiect. După acest pas, la obiectul cu numele *SessionContainer*, trebuie atașată componenta cu numele *InputDataContainer*, cum se poate vizualiza în Figura 9.

După ce se realizează acest proces, informația în format .json ce descrie datele de intrare ale jocului trebuie introdusă în câmpul *InputData* al componentei *InputDataContainer*, urmată cu apăsarea

butonului *Update parameters* pentru actualizarea informațiilor interne ale componentei. Orice modificare a informației în format .json necesită actualizarea textului din câmpul *InputData* și apăsarea butonului *Update parameters*.

După ce se actualizează informațiile interne ale componentei, apar în interfața ei un set de câmpuri specifice datelor de intrare descrise în format .json. Acestea sunt vizibile în Figura 9. Valorile acestor câmpuri se pot modifica în situația în care editorul de Unity nu este în *Play*. În momentul în care se pornește jocul în editor, valorile sunt transmise automat în metoda *OnInit(),* implementată în componenta atașată obiectului cu numele *SessionContainer*. În timpul execuției jocului în editor, se pot modifica doar parametri ce au specificată informația *runtimeUpdate*. Valorile stabilite în timpul execuției jocului în editor nu se păstrează după încheierea execuției.

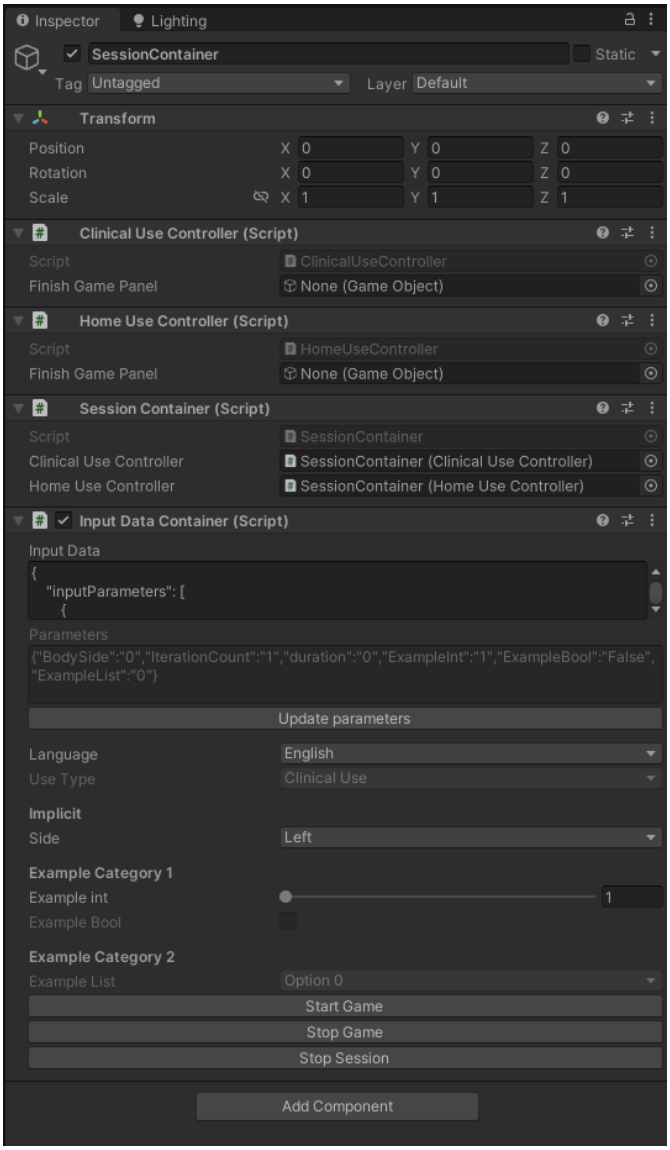

Figura 9. Fereastra de inspecție a componentelor atașate obiectului *SessionContainer*. Se poate observa că acest obiect are atașată o componentă cu numele *InputDataContainer*, ce conține o interfața de editare a valorilor pentru parametri de configurare ai jocului

Bibliografie

[1] Metoda *SendMessage. Documentatie Unity. Online:* https://docs.unity3d.com/ScriptReference/GameObject.SendMessage.html