ARTIFICIAL INTELLIGENCE LIBRARY FOR HTML5 BASED GAMES: DignityAI

Article · September 2016

DOI: 10.16984/saufenbilder.22281

2 authors:

Berkan Uslu
Havelsan
4 PUBLICATIONS **1** CITATION
SEE PROFILE

Ecir Ugur Kucuksille
T.C. Süleyman Demirel Üniversitesi
51 PUBLICATIONS **383** CITATIONS
SEE PROFILE

SAKARYA ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ DERGİSİ
http://dergipark.gov.tr/saufenbilder

Artificial intelligence library for html5 based games: DignityAI

Berkan Uslu[1], Doç. Dr. Ecir Uğur Küçüksille[2*]

21.04.2016 Geliş/Received, 08.08.2016 Kabul/Accepted

doi: 10.16984/saufenbilder. 22281

ABSTRACT

Today, acceleration of internet and common use of web pages, revealed the necessity of work with any browser smoothly for each application without of requirement of any plug-in. Generally, HTML5 is a new body of standards which is formed with the combination of CSS and JavaScript. In this context, by analysing game engines developed for HTML5, their features and advantages are investigated. Although, these game engines are close to catch up with the level of popular game engines, it is seen that none of artificial intelligence library was developed for HTML5 based games up to now. In this study, DignityAI artificial intelligence library is developed to fill this deficiency. Developed library has ability to be integrated to all HTML5 games independently from game engine and to add artificial intelligence dynamics to these games.

Keywords: HTML5, WebGL, Artificial Intelligence Library, DignityAI, Computer Games.

Html5 tabanlı oyunlar için yapay zeka kütüphanesi: DignityAI

ÖZ

Günümüzde internetin hızlanması ve web sayfalarının yaygınlaşması, her uygulamanın her tarayıcıda herhangi bir eklenti gerektirmeden sorunsuzca çalışması gerekliliğini ortaya çıkarmıştır. HTML5 genel olarak, CSS ve Javascript'in birleşimi ile oluşan yeni bir standartlar bütünüdür. Bu noktada, HTML5 için geliştirilen oyun motorları incelenerek; özellikleri ve sundukları olanaklar araştırılmıştır. Bu oyun motorları; popüler oyun motorlarının seviyesini yakalamaya yakın olmalarına rağmen, herhangi bir yapay zekâ kütüphanesinin şimdiye kadar HTML5 tabanlı oyunlar için geliştirilmemiş olduğu görülmüştür. Bu çalışmada, söz konusu eksikliği gidermek amacı ile DignityAI yapay zekâ kütüphanesi geliştirilmiştir. Geliştirilen kütüphane, oyun motorundan bağımsız olarak tüm HTML5 oyunlarına entegre edilebilen ve bu oyunlara yapay zekâ dinamiklerini katabilme yeteneğine sahip bir kütüphanedir.

Anahtar kelimeler: HTML5, WebGL, Yapay Zeka Kütüphanesi, DignityAI, Bilgisayar Oyunları.

[1] Süleyman Demirel Üni., Mühendislik Fak., Bilgisayar Müh., ISPARTA – berkan.uslu@gmail.com
[2] Süleyman Demirel Üni., Mühendislik Fak., Bilgisayar Müh., ISPARTA – ecirkucuksille@gmail.com
* Sorumlu Yazar / Corresponding Author

1. INTRODUCTION

HTML5 and WebGL are among the most popular and latest technologies which are especially focused by the World in recent years. Also, with each research on game developing field, which became stronger by accelerating, increases its invaluable position. Besides, the topic of artificial intelligence, which the large technology companies interested in, perhaps can be shown in the first place in addition to the rising popularity of this valuable research subjects. This situation has gained even more importance by the help of personal assistant with artificial intelligence published by three technology giants such as Apple, Microsoft, and Google.

In this research, first of all, the development of artificial intelligence systems in games and current artificial intelligence libraries are described by considering the history of artificial intelligence in computer games. During the research, all games that can be integrated with the artificial intelligence engine library were analyzed independent of any game engine. However, it was observed that there was no such library that can be used for games based on HTML5. In this point, developed DignityAI library have the feature of being first of its kind. Furthermore, it was observed that each developed artificial intelligence library focuses on a particular type. In this context, the investigations are carried out on the relationship between artificial intelligence and game types. It is centered upon the action (FPS, TPS), role-playing game (RPG), real- time strategy (RTS), adventure, platform, sports and racing from game types. The use of methods of finite state Machine(FSM), hierarchical artificial intelligence, fuzzy state machines (FuSM), behavior trees, path- finding, data driven systems and fuzzy logic artificial intelligence has seen and some information is given.

Subsequently, information about the development of HTML5 and WebGL is given. Then, HTML game engines are examined, the stable and current most popular game engines, which can develop both 2D and 3D games, are handled. Investigated game engines include Construct 2, ImpactJS, PixiJS, PlayCanvas, Three.js, Phaser, Kiwi.js and enchant.js. The sample game in this study is developed by using PlayCanvas game engine.

Finally, information about general structure and all classes of DignityAI artificial intelligence library within the scope of such study is given. In addition, a small game developed by using this library is presented at the end of the study.

2. LITERATURE REVIEW

Wexler, in his work, firstly made an introduction to the history of computer games and examined the development of the artificial intelligence in the computer games. Although the main scrutinized issue was artificial intelligence in computer games, he studied the artificial intelligence system of Black and White, Lionhead Studios' game, and where can the artificial intelligence come in games [1].

Middleton, in his work, observed the history of artificial intelligence in computer games and development of artificial intelligence systems in the last 25 years of history. In the study, he made detailed examination and comparison from chess artificial intelligence designed by Alan Turing in 1950's to Deep Blue computer that defeated Garry Kasparov, from the famous RPG game Dungeons&Dragons to another legendary RTS game Dune II, from Age of Empires series to artificial intelligence systems of Valve's most popular game in the FPS genre Half Life [2].

Stall, in his work, proceed through a very natural example experienced by him to explain the finite state machines and described the finite state machines in a very clear manner [3].

Schwab, in his work, made a very detailed investigation on the s details of developing AI (artificial intelligence) game engine from simple components like navigation, decision-making and inputs to detailed research of game genres and components, from the detailed method analysis such as finite state machines, fuzzy state machines, messaging systems to complete AI game development. Some advanced issues such as genetic algorithms, neural networks, fuzzy logic, behavior trees were described particularly in his book [4].

Lubbers et al., in his work, elaborated on details of HTML programming from HTML and XML which is the basic components of HTML5 to presently JavaScript and CSS that comprise HTML5. He also searched intimately all API libraries of HTML5 which was widely used and had not been examined yet, besides with such review it was given useful examples to learn what browsers are supported by which API [5].

Freeman, in his work, scrutinized JavaScript and CSS that constituent the HTML5 and revealed the detailed guide to use the full power of HTML5 by using such technologies. Focusing on HTML programming he mentioned the methods of forms and form verification and after referred the detailed CSS resource [6].

Parisi, in his work, opened the doors of the world's of WebGL and described the formation and history of the WebGL up to today. Primarily making an introduction to WebGL and then by using Three.js game engine he developed a simple WebGL application. He referred the use of WebGL with graphics, animations, 2D and 3D environments for promotion and developed the sample WebGL game [7].

Rabin, in his work, expressed the artificial intelligence method that depends on the level of detail (LOD) which is a new artificial intelligence approach in RPG games. He mentioned it was an effective technique that speeded up the artificial intelligence system together with the adaptation to artificial intelligence of such technique in graphic programming and that helping to reduce the CPU load [8].

Compton et al., in his work, studied on procedural section design, which is one of the most difficult artificial intelligence subjects, in the platform games and developed the new algorithm for section design improving four-level method [9].

Beirne, in his work, handled the artificial intelligence in the racing games and explained the development of tuning and car modification artificial intelligent with their most used models [10].

Moreover, some projects on github.com has been added within the works in this field by referring some parts of artificial intelligence in HTML5 games.
Gordon, in his work, presents us a framework called Javascript State Machine and focused on finite state machines (FSM) [11].

Cowart, in his work, described behavior trees creating with Machine.js project focused on finite state machines (FSM) similarly [12].

Xu, in his work, just focused on path-finding and studied Pathfinding.js project which is one of the best projects that is specialized in this area by hosting several search algorithms [13].

3. ARTIFICIAL INTELLIGENCE IN COMPUTER GAMES

3.1. History of Artficial Intelligence in Games

Computer games arose with the game named “Tennis for Two” in 1958 which was developed by William Higinbotham, who was a researcher in Brookhaven National Laboratory. This first game developed by William, was only possible to be played with an oscilloscope. The first game that can be played on computer was “Spacewar” developed by Steve Russell from MIT. In 1970s, Nolan Bushnell and Ted Dabney, who will later be the founder of Atari, developed the game named “Computer Space” which would be the first video arcade game. In 1980’s, the first 3D game Battlezone was developed by US government for use of military training and 4 years later namely in 1984, Nintendo game console provided computer games to commercially enter to houses by hitting the market. Afterwards, it became a huge industry together with the acquisition of PlayStation by Sony and spread of computer games in personal computers [1].

When looking the history of artificial intelligence in computer games, it can be said it was revealed firstly on board game genres. The artificial intelligence for chess game in 1950s developed by Alan Turing and Claude Shannon, who are the fathers of artificial intelligence, is shown as the first example of it. In 1952, the checker game “The Samuel Checkers-playing Program” developed by Arthur Samuel from IBM is known as the first game to be self-learning and considered within the earliest examples of artificial intelligence [2].

In 1990s, the fictionalizing period of more intelligent systems began with the use of artificial intelligence in computer games and developing processor technology. Many games in “Turn-base Strategy / TBS” genres (Chess, Checker, Go etc.) are considered as the first game genre which the artificial intelligence was used and “Role-Playing Games /RPG” genre follow it. In 1992, with Dune II’s, which is developed by Westwood Studios and known as the first example of the “Real-Time Strategy /RTS” genre, the working requirement of new and faster algorithms on artificial intelligence real-time calculations and decision-making structures appeared.

In 1998, Half-Life game in “First-person shooter /FPS” genre developed by Valve is varied as the one of the innovative games in artificial intelligence. One of the major innovations on Half-Life game is the use of new genre called “schedule-driven state machine” instead of behavior model that is named finite state machine used on artificial intelligent games and composed of limited numbered states, transition between states, actions.

In 1999, the improving requirement of more intelligent “non- player characters” (NPCs) of developed artificial intelligence in RPG, RTS and FPS game genres was noticed, revealing the games such as Age of Empires II and Unreal Tournament and it began the process of coming up today the use of very high level features.

Nowadays, we are moving forward to a system where the navigation algorithms are risen to next levels, where act and tracking sensors responds faster and the game characters that act and think like humans can be programmed. In a computer game to be created in future, it will be possible to create games with super intelligent NPCs in which the artificial intelligent characters take over the game and all of the humanity struggles to beat the artificial intelligent characters in the game.

3.2. Artificial Intelligence Libraries Used in Games

Many different artificial intelligence libraries have been used in games and until now most of them were integrated to game engines and not functional without using the game engine. The main point to study is the artificial intelligence libraries that work independent from game engines. DignityAI developed in this study is designed with completely independent structure from the game engine and can be applied to all game engines.

Kythera artificial intelligence library is a library which is well known in present days and started to be used in large games. Star Citizen is the leading of these games. It gets ahead of other libraries with respect to both usual functions, such as detection, target selection, hiding, group coordination, and dynamic navigation and behavior trees. Kytera, was designed to be a system based on C++ that supports Windows, Mac and Linux with 32-bit and 64-bit processor architecture and to be integrated to all game engines. However, it cannot be integrated with the HTML5.

Havok AI artificial intelligence library is a library that focuses on the main navigation functions like path-finding and path-following. One of its most significant features is to create automatic navigation mesh. In spite of the fact that Havok AI was developed by using C++, it can still give support to all leader platforms. However, there is no integration with the HTML5.

Rain artificial intelligence library is developed for Unity game engine but it is sold and distributed independent of it. It consists of the advanced navigation features like automatic navigation meshing, path-following and path finding. However, it has no HTML5 support. WebGL export option was improved with Unity 5 but it doesn't give native HTML5 output.

Kynapse artificial intelligence library is a large artificial intelligence library which is developed by Autodesk Company and nowadays separated to two products named Gameware Navigation and Gameware Cognition. Gameware navigation is the artificial intelligence library searching for options to the requirements of usual path-finding and navigation. Gameware Cognition is a tool that helps to create behavior trees supporting the visual programming platform.

PathEngine artificial intelligence library is the artificial intelligent library that focuces on the best path-finding algorithms that is improved until now. PathEngine which has a very well designed action model with the features of dynamic navigation mesh creation, obstacle recognition, crush monitoring proved itself by being used by many big projects.

Masa Life artificial intelligence library is an artificial intelligent library that focuses on decision-making. Masa Life which is developed with C++ that works on Windows and is a small integration of Unity Windows integration, approaches mostly to subjects such as decision-making mechanisms, behavior trees and navigation. In addition to this, it has the working fields like reasoning, information provision and processing.

Cyntient artificial intelligence library is an artificial intelligent library developed for the game industry. It is a system that provides virtual characters' to analyze by learning each other's behaviors and to take action. The aim of Cyntient is to develop an open world space game named Galak-Z through providing a realistic experience by creating intelligent and emotional characters.

As it mentioned above, there are many artificial intelligence libraries and additional to these there are integrated artificial intelligence modules inside the game engines. But, because none of these systems can be integrated to technologies such as HTML5 and WebGL at one point they are platform dependent and require advanced systems to improve. In this manner, improved DignityAI library contains many classic artificial intelligent items such as path-finding, decision-making, behavior and task managing that focuses on HTML5 and WebGL.

Dignity AI is completely open source coded and with this aspect it differs from other artificial intelligence libraries. Besides, it is aimed to be use in all browser based systems with the power of HTLM5 and designed in architectural structure to be used for nearly all game genres with powerful basis.

3.3. Artificial Intelligence Relationship with Game Genres

Most of game engines designed until now carried out their works following the way based on to develop games belong to specific a genre and to fulfill certain functions within these games. One of the major factors in the issue being specific to genre can be summarized with that the games are very complex compared with the other systems and game and/or game engine developers focuses on the ways to solve the problems by dividing them into pieces.

Game genres customized the artificial intelligence libraries while shaping the game engines. Such an extent that there are unique different mechanics of each game. In following each paragraph, the relationship between artificial intelligence and game genres are described.

Action (FPS, TPS) games are generally developed in game genres of First Person Shooter (FPS) and Third Person Shooter (TPS). The path finding property is one of the main mechanics of action games. The other property is

EnemyAI term it can be called enemy artificial intelligence. In this type of action games, there usually is an enemy that can act differently in certain intelligence levels. Behavior trees must be present for these behaviors. They must know what kind of response they should give against which kind of events and their decision-making mechanism must have been developed. Since the action games were designed for single player, all other items except the player must have artificial intelligence. It is one of the most widely used game genres of artificial intelligence with this aspect.

Role-playing games (RPGs) is emerged as the game genres that have existed since the first release of computers. RPG games can be described as which is played with huge maps and the level reaching of the player by performing certain duties. However, although this game style is defined as simple as this, it evolved into very complex games and by including many mechanics and moving to online side as well it created MMORPG (Massively Multiplayer Online Role Playing Games). The artificial intelligence in RPG games begins with the path-finding of controlled character and going from one place to another. Inventory list of the character exists, many various types of materials can be found in this inventory list such as weapons, foods, clothes. The character learns with trial which weapon to be used when facing against the enemy to be killed, and also enemy can have weapons and spell power. However, these characters can generally manage the basic actions like using weapons and detection by hosting simple behavior trees. The main feature that separates RPG type games from the other types is the multitude of interactive objects.

One of the artificial intelligence techniques used in RPG games is a method named artificial intelligence according to level of detail (LOD) and used inside the games that occurs on huge maps like Baldur's Gate series. CPU overuse can be hindered by using this applied technique preventing the operation of the artificial intelligence of the objects away from the character. Thus, faster and more stable artificial intelligence systems can be designed [8].

Real-time strategy games (RTS) is perhaps one of the game types where the artificial intelligence used most widely. While any army, city etc. that is managed by strategy is controlled by player, items such as the army, city are managed by artificial intelligence. RTS games are one of the most difficult games to develop with regards to artificial intelligence. Because CPU must graphically process the items like characters that are moving and buildings that are formed while it is busy with the artificial intelligence. Additional to optimization difficulties, it is very detailed in terms of diversity of artificial intelligent items. In this game genre, the finite state machines is frequently used for static tasked units. It profits from behavior trees and path-finding abilities. Nevertheless, the structure named Fuzzy State Machine (FuSM) was used in modeling of strategic units. Fussy state machines have the configuration that make a decision by calculating various situations inside the system. Another case that is used in RTS game genre is the structure named the Hierarchical AI. This structure can be illustrated as follows: when attacking to somewhere it is necessary to fight other enemies that is faced even if they are not the main target. In this case, the new task will be taken hierarchically and it is proceeded to the main target after sorting out.

Adventure games is type that is still kept alive today with Walking Death series and played much despite being a very old kind. In action games, it is not desired players to follow a certain flow, it is required to explore the field and to reach next level by gathering or finding the hidden objects around the field. Adventure games involve the general enemy artificial intelligence, detection systems and classical behavior trees.

Platform games are the game genres that stand the test of time since the earlier times of computer games and has again come up with the developed mobile technologies. Platform games generally do not have an item as path-finding. It is the simplest and least complicated game type with regard to artificial intelligence.

The most complicated artificial intelligence used in platform games is found in map designs of games. Procedurally the creating of stages in platform games is usually done by artificial intelligence. Moreover, the process is even more difficult from the design of stages in strategy games and RPG. Since because changes are so few, a clever artificial intelligent algorithm required in order to design the objects that does not repeat each other and to eliminate the monotony [9].

Sport games are one of areas that the artificial intelligent used most heavily. Artificial intelligence is very complex in sport games; it consists of set of rules and team calculations about applied sport branch. A complex artificial intelligence based on sensors, detection and certain behavior trees is found in sport games. Furthermore, it is required to contain the tactical artificial intelligence for situations such as changes to be done with the progress of the game and the shifting of the tactic system, etc. Additional to use of finite-state machine fuzzy-state machines and data-driven systems are used frequently. The reason is that the data will change during the games and the parameters like performance of player and condition will be monitored and processed by state machines.

Racing games have their own type of artificial intelligence mechanics. The artificial intelligence in racing games can be classified as track display, control and detection the lines of other racers, the control of artificial intelligent character's race car (engine, car, plane, etc.). Moreover, the innovations with artificial intelligence-controlled car modification and tuning can be seen in this area [10].

3.4. HTML5 and WEBGL

HTML is one of the oldest text editing languages, whose roots almost goes back to internet and used since 1993. HTML used 2.0, 3.0 and 4.0 versions until 1999 and lastly 4.0.1 version in 1999. These HTML versions were developed by World Wide Web Consortium (W3C) organization's group called the HTML Working Group. Later, stopping their works on HTML, the group diverged to another web standard XML and then XHTML accordingly. Afterwards, a small group of people that wanted a new web standard, gathered under the name of Web Hypertext Application Working Group (WHATWH) in 2004 and published the HTML5 specifications [5].

After these researches, W3C began to work about HTML5 in 2006 and the first operational version was released by them in 2008. HTML5 is a new body of standards which is formed with combination of HTML5, CSS and JavaScript. HTML5 is used with tags and new elements, CSS is used with the visual parts of these tags and elements, JavaScript is used to process the content of all structure and to response the actions of user and to utilize programmatic advantages of HTML5 [6].

HTML5 was published as a standard by W3C on September 28th, 2014. HTML5 brings some new features like Canvas (2D and 3D), Cross Document Messaging, Geolocation Audio and Videos, Forms, SVG, WebSocket API, Local Storage, Offline Web Apps, Drag and Drop, Web Workers, Servers-Sent Events, XMLHttpRequest Level, Local Storage, Offline Web Apps, Drag and Drop, Web Workers, Server-Sent Events, XMLHttpRequest Level 2.

System has become even stronger by developing several API on HTML5. The most important one of these is display of 3D visual through WebGL without requirement of any plug-in installation on browser side. WebGL is essentially OpenGL's portion adapted to HTML5. So, HTML5 can create 3D objects with JavaScript without the necessity of use of any other programming language or can perform animation by importing a character model.

Developers can use all hardware graphic processing power of computer via any browser with WebGL. Before the WebGL developers was able to use hardware graphic processing and other features with games they developed by making users to install a plug-in (such as Adobe Flash Player) or providing them to download and install the application special to own operating system (exe, app, deb, apk, etc.) to their devices [7].

Khronos Group was developed the WebGL similar to other HTML5 APIs by building it on OpenGL ES 2.0. 3D graphic applications became able to use in the regular html elements and low-level DOM interface through this API. Additional to this, 3D web applications can be created just as upper-level computer games can be developed. The other yield of integration of WebGL on OpenGL ES is to operate on low-power devices and mobile platforms (iPhone, iPad, Android Devices, etc.) with the correct resource management in all leader mobile operation systems by virtue of the fact that OpenGL ES is adapted for embedded systems. However, WebGL is a very low-level library like OpenGL. The use this library requires expertise and is as complex as OpenGL but the use of WebGL has become simpler through some game engines. The best example of this case can be given as PlayCanvas, Three.js and Babylon.js. By these engines, 3D games can be developed just like modern game engines Unity, Unreal Engine, CryEngine and all processes can be performed via an internet browser.

3.5. HTML5 Game Engines

There are two options about game developing with HTML5. The first is Canvas where 2d games can be developed, the second is WebGL where the 3D games are developed. Both 2D and 3D games can be developed under the same platform via game engines. HTML game engines are briefly described in the following paragraphs.

Construct2 game engine developed originally as hobby by a group of student in 2007 and emerged with the name of Construct Classic. Construct 2 was released HTML5-driven in 2011 and many radical changes made on this game engine. It embodies very important features such as multi-platform support, easy learning and visual programming [14]. Construct 2 was designed for development of 2D games and it was mentioned the users can develop game without any knowledge of programming. SDK Template file offered by Construct must be downloaded in order to use DignityAI library inside Construct 2. It can be possible to easily add artificial intelligence features to elements in the game engine through the codes to be transferred into this template.

ImpactJS is one of the popular and paid game engines that was begun to develop by Dominic Szablewski in 2010. One of the biggest advantages of this game engine has Entity structure. It effects the whole by calling update and draw methods in all Entity structures that are connected to draw() and update() methods, which located in the engine.. /libs/game/entities/ directory can be found inside the project, which the Dignity AI will be used. When DignityAI 1.0.0 version is added to this directory at https://github.com/berkanuslu/dignityai/releases/download/v1.0.0/dignityai.v1.0.0.min.js address, artificial intelligence features can be added to intended objects by using <script src="libs/dignityai.v1.0.0.min.js"></script> in index.html page [15].

PixiJS is a game engine which is built by Good Boy Digital, build on Node.js and distributed free of charge through Github. Contrary to other 2D game engines, PixiJs

renders the game with the WebGL to increase the performance if the browser has WebGL support. However, it carries on the operation via Canvas with the browsers which have no WebGL support. PixiJS is showed as a tool in render parts of most game engines like ImpactJS, Phaser, PandaJS. By creating a /libs/ folder in the project where the DignityAI will be used and when DignityAI 1.0.0 version is added into this folder at https://github.com/berkanuslu/dignityai/releases/download/v1.0.0/dignityai.v1.0.0.min.js address, artificial intelligence features can be to intended objects added by using <script src="libs/dignityai.v1.0.0.min.js"></script> in index.html page.

PlayCanvas differently from other game engines allows developing the game with a Cloud account on web. Codes can be modified by the help of the editor named PlayCanvas Code Editor. One of the most important properties of PlayCanvas is that it has its own level editor. This editor helps to design the 3D scene and to edit the materials just as big game engines (Unity, Unreal Engine etc.). Another feature that makes PlayCanvas different from other game engines is that the users can use all public projects inside their own project repository by forking. Installation process is completed when DignityAI 1.0.0 version file, which is located in https://github.com/berkanuslu/dignityai/releases/download/v1.0.0/dignityai.v1.0.0.min.js address, is downloaded and when this file is linked to an Entity. By this means artificial intelligence features can be added to intended objects.

Three.js might be an engine distributed free of charge and focused on WebGL and took the first step for developing 3D applications with WebGL. Many renderers can be found in Three.js such as WebGL, Canvas, SVG, CSS3D, DOM. It allows many items such as stage, camera, animation, lighting, material and shader. >An editor design is making in Three.js like PlayCanvas and it is planned to realize 3D scenes with the help of this editor. By creating a /libs/ folder in the project where the DignityAI will be used and when DignityAI 1.0.0 version is added into this folder at https://github.com/berkanuslu/dignityai/releases/download/v1.0.0/dignityai.v1.0.0.min.js address, artificial intelligence features can be added to intended objects by using <script src="libs/dignityai.v1.0.0.min.js"></script> in index.html page.

Phaser is a HTML5 game engine which is distributed free of charge and published as open source. Phaser is concentrated on 2D game developing side. Phaser, which uses PixilS on render side, allows to develop 3D applications by supporting the HTML5 and WebGL. It includes many features such as preloader, physical system, sprite, animation, particle, input, sound, tilemaps, scaling to devices, mobile browser support [16]. There is plug-in logic in Phaser. All features added to application is developed with the help of these plug-ins. The sample plug-in file that can be found in the https://github.com/photonstorm/phaser-plugins/blob/master/SamplePlugin/SamplePlugin.js address is already similar with the DignityAI developing structure. By creating a project inside phaser plug-ins and then transferring DignityAI files into this folder, DignityAI will be suitable for this game engine as well.

Kiwi.js is an open source game engine which was designed to develop mobile and desktop game on HTML5. It has WebGL support for 2D and 3D design features. It benefits from CocoonJS platform to publish the games [17]. It supports the plug-in system similarly to most of games engines like Phaser. By creating a /libs/ folder in the project where the DignityAI will be used and when DignityAI 1.0.0 version is added into this folder at https://github.com/berkanuslu/dignityai/releases/download/v1.0.0/dignityai.v1.0.0.min.js address, artificial intelligence features can be added to intended objects by using <script src="libs/dignityai.v1.0.0.min.js"></script> in index.html page.

Enchant.js is a considerably simple JavaScript library. It is used to develop game and application. It is developed by researchers Akihabara Research Center, Tokyo, at first in 2011 and shared as open source with the license of MIT. Because of its simple and plain framework, beside game developing it is frequently used for application developing. It supports all design methods like Canvas, DOM and WebGL. By creating a /libs/ folder in the project where the DignityAI will be used and when DignityAI 1.0.0 version is added into this folder at https://github.com/berkanuslu/dignityai/releases/download/v1.0.0/dignityai.v1.0.0.min.js address, artificial intelligence features can be added to intended objects by using <script src="libs/dignityai.v1.0.0.min.js"></script> in index.html page [18].

Many more HTML5 game engines are being developed and added to this list everyday. Several game engines such as Babylon.js, GameMaker, Turbulenz, Cocos2d-X, Isogenic Engine, Panda.js, Crafty, voxel.js, MelonJS, stage.js has been designed for the ones that want to develop games with HTML5. When looked into these game engines and APIs, it can be seen that lots of items like graphic, audio, video, physics are thought and they can already be used in HTML5 based games. However, it is known there is no game engine or API which hosts any library about artificial intelligence. DignityAI is developed in order to complete the missing parts at this point.

4. DEVELOPED LIBRARY: DignityAI

4.1. General Class Structure of DignityAI

Class diagram of the developed library is given in Figure 1. DignityObject is placed at the top when looked at the

DignityAI general class structure. All the other classes are fundamentally inherited from this class. If we are to give a short information about the classes, what are the functionalities of all the classes and what they are used for will be seen in general terms.

4.2. DignityPath

DignityPath class is the class that handles the basic path-finding. It is specifically designed as to cover the navigation functions both for 2D and 3D games.

All the items inherited from DignityAIBase type has sets of DignityPath by the name of a paths. This sets the paths which the artificial intelligence character will follow in relation with its tasks. More than one path can be defined within the tasks. Defining of the DignityPath class in the DignityAIBase, provides that it can reach previous paths through any task on DignityMission. Thus, any task transforms into a more flexible structure.

Structure is developed in this way by flexible design. For the path given in the DignityPath class, together with the start and finish locations other sub paths which will be used to move to those start and finish locations can be defined. Also, whether the given path will or will not repeat from end to start and move on a random location can also be determined through the same class.
Statements defined by this class are actually used by a move() method for DignityAIBase class and classes inherited from this class. This method provides the object to get into act to follow the path designated in the task definition by reading the paths on the artificial intelligent object. Fundamentally it is mandatory to determine and to fill the finish point; at the end the object is moved towards the finish point. However, some basic algorithms are used for the object to calculate the path it will take and draw a route according to the obstacles on the way. There are many different search algorithms such as A*, IDA*, MA*, Breadth-First-Search, Best-First-Search, Dijkstra, Jump Point Search and many more within these algorithms. From these algorithms, DignityAI uses the most used and most common A* algorithm. However, all the other algorithms or a library such as Pathfinding.js which has a HTML5 library containing all these algorithms can also be easily integrated into DignityAI.

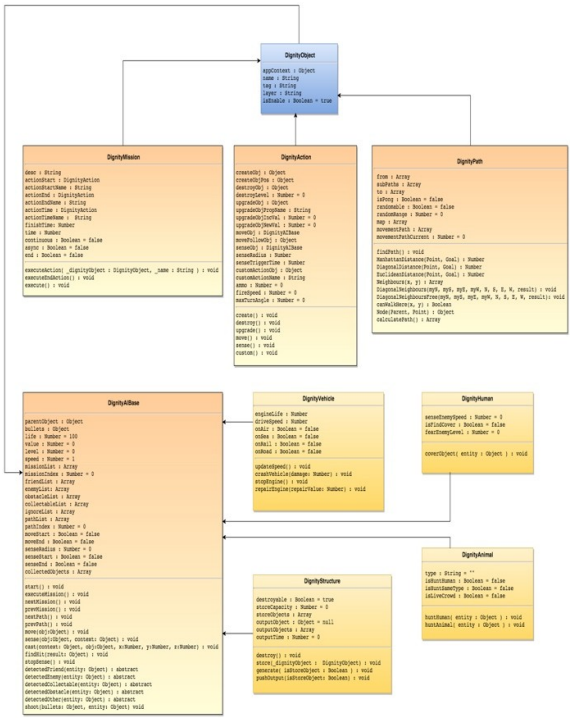

Figure 1: DignityAI Class Diagram

4.3. DignityAction

DignityAction is actually a process part of the DignityAI. All the actions are defined within DignityAction. DignityAction has move(), sense(), destroy(), create(), upgrade() and custom() main functions.

With move() function; the pathList member defined in the DignityAIBase follows the path which is defined on the DignityPath class' object. It finds the shortest path for this process and creates an action. Here the basic condition is; to provide the object, which will be moved with the moveObj property defined in the DignityAction class, to trigger the move method defined in DignityAIBase.

With sense() function; detection according to enemy lists (enemyList) and friends lists (friendList) defined in DignityAIBase is done and main processes are performed according to these detections. This method also provides the object, given to senseObj property and detection to be done, to trigger the sense method defined in DignityAIBase.

With destroy() function; object pointed as target, which is defined with destroyObj property and is from DignityAIBase class or from another class inherited from this, gets damaged. This damage, reduces life feature by the value of destroyLevel for the objects inherited from DignityAIBase class.

With create() function, any object defined with createObj property are created on stage. Also, with the function defined in DignityAction named createObjPos position where the object will be created is given.

upgrade() function is used to increase all the values of the object defined with upgradeObj property. In any way processes, such as level upgrade, health increase are done by this function. For this, name of the object to be upgraded in upgradeObjPropName is given in String. With the upgradeObjIncVal property, information of how much the property, which is defined with upgradeObjPropName, will be increased is given. Thus, given increase value adds to that value. In case a negative value is given, it means that parameter's value is decreased. However, in case a value assignment to be done directly but not as increase/decrease the value of the variable, upgradeObjIncVal property shall be given 0 (zero) and new value of the variable is given to upgradeObjNewVal. An object's value assignment to another can be done by this way.

In the basic custom() function makes required functions of all the objects', which are not defined in DignityAction class, run as actions. It means in case another action apart from move, sense, destroy, create and upgrade actions, custom() function shall be used. This function calls the method, which is given with customActionObjName, of the object' which is given with customActionObj, as action.

4.4. DignityMission

DignityMission class is a class provides the mission managing and can assign mission to each artificial intelligent character. Elementarily is has a structure where missions can be defined as lists, actions for the missions can be defined by DignityAction and these actions can be run as synchronously or asynchronously.

Essentially DignityMission consist of description and action definitions. While description is defined for the cases in which the mission will be shown to users, action is defined by an object assignment in DignityAction class. For this actionStart, actionEnd and actionTime properties are defined in DignityMission class. By means of these properties a start action, end action and a timing action which can be run in any time can be defined. Methods to be used by actions are decided by DignityAction class method names (move, sense, create, destroy, upgrade and custom) which are defined in actionStart, actionEnd and actionTime properties. Also, whether the action will run synchronously or asynchronously is determined by a property named async.

Control of the mission's end in the DignityMission is provided with finishTime variable. If finishTime variable is given 0 (zero) it means the mission will be provided by temporary actions. If any value is given to this variable, mission has a time to live in terms of seconds. actionEnd action runs at the end of this time and comes the next mission. Another parameter from the mission transition is async variable. By means of this variable, it is decided whether the mission will be synchronous or asynchronous. All the asynchronous missions are run at the same time. But in case the first mission is synchronous, second and third missions are asynchronous, then the first mission runs primarily and then second and third missions are run at the same time after the first mission is finished.

4.5. DignityAIBase

DignityAIBase class is the basic class of the artificial intelligent objects. All the artificial intelligence characters are characters inherited from this class. Classes such as DignityStructure, DigntiyHuman, DignityVehicle and DignityAnimal are the classes to be used for the artificial intelligent characters customized on DignityAIBase and artificial intelligent characters inherited from this class. Also, as much as customization can be done by adding objects inherited from DignityAIBase.

All the previous classes are classes used to improve the DignityAI basic mechanics and to develop the basic properties to be used in DignityAIBase. Basic use area of these classes is DignityAIBase and classes inherited from it.

Properties named speed, life, value and level are defined in DignityAIBase class. These are the common properties of all the artificial intelligent objects. Together with them artificial intelligent objects must have a mission list (missionList). This mission list starts to run respectively (asynchronous ones are at the same time) to run the mission list. In any case if it comes to a synchronous mission, finish of this synchronous mission is to be waited before running the other asynchronous missions.

Also, there are properties name friendList, obstacleList and collectableList. With these properties detection processes in the sense() method, which is within the DignityAction classes' main methods, are done. sense() function realizes the defined detection processes of the user by looking at these lists and as a result of the detection process detectedFriend(), detectedEnemy(), detectedCollectable() and detectedObstacle() methods are called. Besides, unwanted objects detection can be determined with the ignoreList property. Objects in the ignoreList are not subject to object detection process in any way.

Also, pathList array in the DignityAIBase class, by holding the object in DignityPath type it provides the use of these paths. If there are no path lists added to this artificial intelligent character, it provides it not to be moved to a newly added path on the system. By this way,

access to a single source through all the other classes is provided.

executeMission() method is defined in order to run any existing mission in DignityAIBase. Also nextMission() and prevMission() methods are defined in order to navigate between missions. Likewise for navigation processes in pathList property nextPath() and prevPath() methods exist in order to use in move() method.

There is move() method in DignityAIBase, which uses the DignityPath class for moving the artificial intelligent objects and in order to regulate the objects movements to coordinate points within the existing path. This method respectively provides the movement of the object to all the points in the calculated shortest way of the path data and when the movement process is finished (when reached to final point) to end the action or to move forward to next point if exist.
For detection processes of the artificial intelligence objects, sense() method is defined. In this method, essentially rays are sent from the objects position to the positions in the scanning area by using classic Ray Cast method. cast() method is defined for sending ray process. Objects that hit with the ray (in order to object to detect this hit process object need collision component) falls into findHit() method as the result. And in this method, by looking that the object is not in ignoreList but in which one of the friendList, enemyList, obstacleList, collectableList that list's detection methods triggered. If the found method is in ignoreList then other lists are not checked.

Also, for the other objects, which are inherited from DignityAIBase class, to interpret the detection statuses in different ways, when an object, which does not fall into any of the lists above, is detected detectedOther() method is defined in order to be able to reach these objects and to establish a flexible structure at the detection point. This method is called if the detected object is neighter in the ignoreList nor in any of the other lists. Customizations can be done for the objects inherited from DignityAIBase class and override this method.

Last defined method is the shoot() method. According to the detection result, desired objects can be fired at. Objects to be fired (bullet, cannon, bomb, arrow, etc.) are stored in file named bullets.js. Different weapons can be determined according to each object.

4.6. DignityStructure

It is designed to add artificial intelligent features to fixed objects such as building, structure which are inherited from the DignityAIBase. By this class mission and action can be given to a structure. Normally objects such as buildings and structures do not move; but still to moving feature can be added to these structures by using move() action.

There are some customized properties within the DignityStructure. First of these properties is to define the structure to be collapsible, destroyable. Second property is the capacity of the structure. This capacity indicates the object capacity which it can hold, hide inside. If wanted, this capacity can be increased by multiplying with the level feature in DignityAIBase class. Most important property of the structures is them to have a mechanism which gives an output in a given time. By this means they can produce defined objects in given periods and store the outputs if required.

4.7. DignityVehicle

It is developed to create a vehicle model with artificial intelligence. There is a drive() function within this class, which uses the move action of DignityAction class however managing the movement situation specific to the class. By means of this method vehicle's speed is found by calculating the engine condition and driving speed. When the engine condition goes bad, speed reduces automatically. When the vehicle crashes with another vehicle, engineLife property gets reduced by the valu of the crash' intensity and speed is calculated again according to this. crashVehicle() method controls the status of crash. There is a repairEngine() method for improving the engineLife value when received an object such as repair kit. Lastly there is a stopEngine() method to stop the vehicle on demand.

4.8. DignityHuman

It is defined to create an artificial intelligence human model. It additionally contains features such as detecting the enemy, to search hiding spots for warrior characters and move towards this spot, getting scared level from enemies.

4.9. DignityAnimal

It is a class designed just as DignityHuman and customized according to animalistic properties and able to create animal objects. DignityAnimal class additionally has the information of animal types, whether it will hunt down the humans or not, whether it will hunt down its own kind or not and whether it lives in packs or not. In the light of these information; detecting humans and hunt them, detecting its own kind and hunt them and also detecting its own kind and go next to them situations are built.

5. A SAMPLE GAME DEVELOPED BY USING DignityAI

When programming a sample game by using DignityAI rather than a visually complete prepared and production oriented structure, a structure which uses the basic mechanics of DignityAI, and completes missions on a

simple map and operates necessary actions as required by using the DignityAIBase and DignityAnimal, DignityStructure, DignityVehicle classes inherited from DignityAIBase was built. Also, while developing the sample game boundaries of the DignityAI library and extra properties, which can be developed, are set to be included in the later studies.

Game to be developed shall define the paths and obstacles on a randomly created 48x48 grid map. Afterwards, a simple tank model is controlled by DignityVehicle class and all the properties such as finding path, detection, mission managing, action managing are handled. Besides that, there is a DignityStructure object named RandomModelCreator which inserts objects that can gather at random points on the map. Lastly there are RedCreator and BlueCreator objects which are DignityStructure objects and produce Red and Blue teams. These objects create tanks on stage periodically and provide these tanks to follow the determined path by assigning coordinates on the random map and meanwhile making them to show different acts against the enemies and collectible objects that they detect.

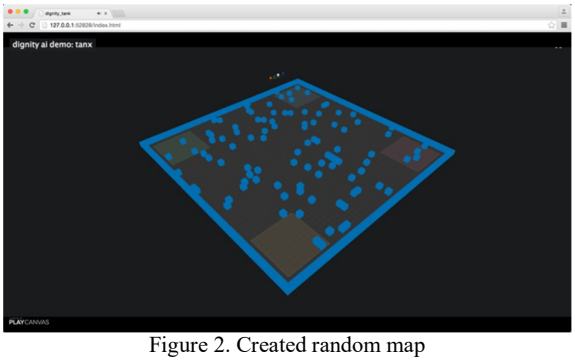

Figure 2. Created random map

As can be seen in Figure 2 the map will be created randomly every time the page is refreshed. After the map is created, object named RandomModelCreateor is created and first mission definitions are made in order to create objects that can gather on random points on the map.

```
addRandomModelCreator: function() {
//create random model creator
var _action = new DignityAction();
_action.customActionObj = this;
_action.customActionName = 'createRandomModel';
var _mission = new DignityMission();
_mission.async = true;
_mission.continuous = true;
_mission.actionTime = _action;
_mission.actionTimeName = 'custom';
_mission.time = 10000; //create every 10 seconds
var _creator = new DignityStructure();
_creator.missionList.push(_mission);
_creator.start();
},
createRandomModel: function() {
var model = this.getRandomModel();
model.setPosition(this.getRandomPosition());
},
```

By means of addRandomModelCreator() method, firstly a DignityAction object is defined. createRandomModel() method is defined to this object's customActionObj and customActionName properties, which are in the same script. After the action definition, a DignityMission object is created to manage how this action will work. This object contains a mission, which works asynch and continues constantly. It calls the DignityAction's custom() method in every 10 seconds. And this method also calls the above described relevant method and the model is created in a random point. Figure 3 shows models randomly created on the map.

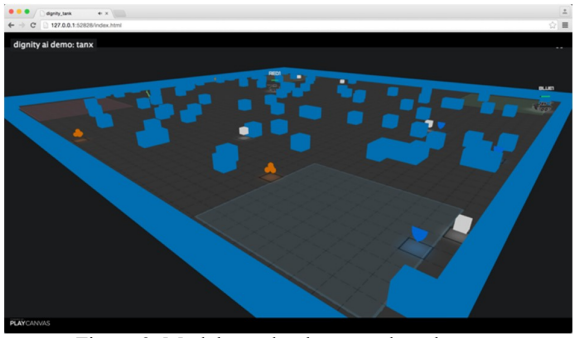

Figure 3. Models randomly created on the map

Together with these one of the 4 models will be randomly added on the map in every ten seconds. After finish the creation of these random creations, it is continued to create tanks one each for red and blue team and to define the artificial intelligence objects and their functions. Process of creating a tank for the red team is as follows.

```
addRedTankCreator: function() {
        //create red tank creator
        var _action = new DignityAction();
        _action.customActionObj = this;
        _action.customActionName = 'createRedTankModel';

        var _mission = new DignityMission();
        _mission.async = true;
        _mission.continuous = false;
        _mission.actionTime = _action;
        _mission.actionTimeName = 'custom';
        _mission.time = 10000; //create every 10 seconds

        var _creator = new DignityStructure();
        _creator.missionList.push(_mission);
        _creator.start();
    },
    createRedTankModel: function() {
        var tankName = 'tank_' + new Date().getTime();
        var newTank = this.tank.clone();
        newTank.setName(tankName);
        newTank.enabled = true;
        newTank.setPosition(this.getRandomPositionRange(2, 10, 2,
10)); //left bottom
        newTank.addComponent('collision', { type: 'box', halfExtents:
new pc.Vec3(0.5, 0.5, 0.5) });
        this.tanks.addChild(newTank);
        var ourPath = new DignityPath();
        ourPath.map = this.map.script.createmap.map;
        var mapX = Math.round((47.5 - newTank.getPosition().z) / 1.0);
        var mapY = Math.round((newTank.getPosition().x - 0.5) / 1.0);
```

```
        while(ourPath.movementPath.length == 0) {
            ourPath.from =[mapX,mapY];
            ourPath.randomable = true;
            ourPath.findPath();
        }

        console.log('red path: ' + ourPath.movementPath);

        var vehicle = new DignityVehicle();
        vehicle.appContext = app;
        vehicle.enemyList.push('tank');
        vehicle.enemyList.push('damage');
        vehicle.enemyList.push('repair');
        vehicle.enemyList.push('shield');
        vehicle.enemyList.push('default');
        vehicle.ignoreList.push('wall');
        vehicle.pathList.push(ourPath);
        vehicle.parentObject = newTank;

        newTank.script.tank.setHP(10);
        newTank.script.tank.ai = vehicle;
        newTank.script.tank.hidden(false);
        newTank.script.tank.setTankName('red1');

        var firstAction = new DignityAction();
        firstAction.moveObj = vehicle;

        var firstMission = new DignityMission();
        firstMission.async = true;
        firstMission.actionStart = firstAction;
        firstMission.actionStartName = 'move';

        var secondAction = new DignityAction();
        secondAction.senseObj = vehicle;
        secondAction.senseRadius = 10;

        var secondMission = new DignityMission();
        secondMission.async = true;
        secondMission.actionStart = secondAction;
        secondMission.actionStartName = 'sense';

        vehicle.missionList.push(firstMission);
        vehicle.missionList.push(secondMission);
        vehicle.start();
    },
    addBlueTankCreator: function() {
        //create blue tank creator
        var _action = new DignityAction();
        _action.customActionObj = this;
        _action.customActionName = 'createBlueTankModel';

        var _mission = new DignityMission();
        _mission.async = true;
        _mission.continuous = false;
        _mission.actionTime = _action;
        _mission.actionTimeName = 'custom';
        _mission.time = 10000; //create every 10 seconds

        var _creator = new DignityStructure();
        _creator.missionList.push(_mission);
        _creator.start();
    },

    createBlueTankModel: function() {
        var tankName = 'tank_' + new Date().getTime();
        var newTank = this.tank.clone();
        newTank.setName(tankName);
        newTank.enabled = true;
        newTank.setPosition(this.getRandomPositionRange(37, 45, 2,
10)); //right top
        newTank.addComponent('collision', { type: 'box', halfExtents:
new pc.Vec3(0.5, 0.5, 0.5) });
        this.tanks.addChild(newTank);

        var ourPath = new DignityPath();
        ourPath.map = this.map.script.createmap.map;

        var mapX = Math.round((47.5 -
newTank.script.tank.entity.getPosition().z) / 1.0);
        var mapY =
Math.round((newTank.script.tank.entity.getPosition().x - 0.5) / 1.0);

        while(ourPath.movementPath.length == 0) {
            ourPath.from =[mapX,mapY];
            ourPath.randomable = true;
            ourPath.findPath();
        }

        console.log('blue path: ' + ourPath.movementPath);
        var vehicle = new DignityVehicle();
        vehicle.appContext = app;
        vehicle.enemyList.push('tank');
        vehicle.enemyList.push('damage');
        vehicle.enemyList.push('repair');
        vehicle.enemyList.push('shield');
        vehicle.enemyList.push('default');
        vehicle.ignoreList.push('wall');
        vehicle.pathList.push(ourPath);
        vehicle.parentObject = newTank;

        newTank.script.tank.setHP(10);
        newTank.script.tank.ai = vehicle;
        newTank.script.tank.hidden(false);
        newTank.script.tank.setTankName('blue1');

        var firstAction = new DignityAction();
        firstAction.moveObj = vehicle;

        var firstMission = new DignityMission();
        firstMission.async = true;
        firstMission.actionStart = firstAction;
        firstMission.actionStartName = 'move';

        var secondAction = new DignityAction();
        secondAction.senseObj = vehicle;
        secondAction.senseRadius = 10;

        var secondMission = new DignityMission();
        secondMission.async = true;
        secondMission.actionStart = secondAction;
        secondMission.actionStartName = 'sense';

        vehicle.missionList.push(firstMission);
        vehicle.missionList.push(secondMission);
        vehicle.start();
    },
```

When looked into the process blog above addRedTankCreator() and addBlueTankCreator(), which firstly definitions of DignityAction, DignityMission are made, can be seen. Here also a customAction definition is made, a mission definition which will work after 10 seconds but won't repeat is made with DignityMission too. 10 seconds waiting after the opening, createRedTankModel() method is called and artificial intelligence tank character is added to scene as shown in Figure 4.

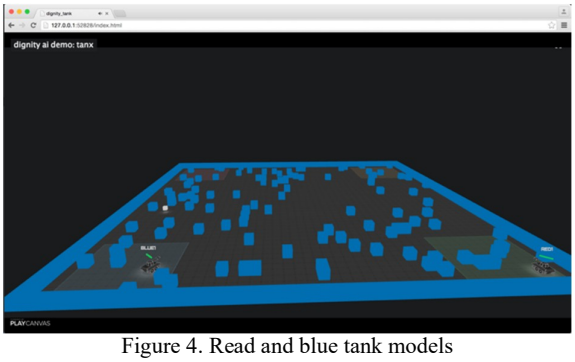

Figure 4. Read and blue tank models

When looked into createRedTankModel method, firstly a tank cloning process is completed. Collision component is described for tank model too. movementPath property gets created by means of findPath() method by linking this object to the created map.

DignityVehicle class is used in order to give artificial intelligence features to tank model. With the help of this class, tank's enemy (enemyList) and friend (friendList) lists are defined. Meanwhile a wall model is added into ignoreList in order to not to detect the wall. Afterwards, previously created DignityPath object is given to pathList as well. Here, parentObject and appContext properties are used to access to game engines's sources.

After the cloning of the tank model, which was created in the game engine and does not have artificial intelligence feature, created DignityVehicle object can use the game engine functions with the ai property in this object.

There are two action and two mission definitions for the tank. First of them is the move action. This action is set to work as asynchronously at the beginning. Second action is the detection function. Here, it is provided for tank to scan a 10-unit area by giving a senseRadius. By adding all these properties to DignityVehicle's missionList, start() method of the DignityVehicle object is called. This allows the activation of the artificial intelligence by running these missions.

After all these processes, two tanks for two different teams detect the enemies and fire at them with the shoot() method as in Figure 5 while moving on randomly given paths.

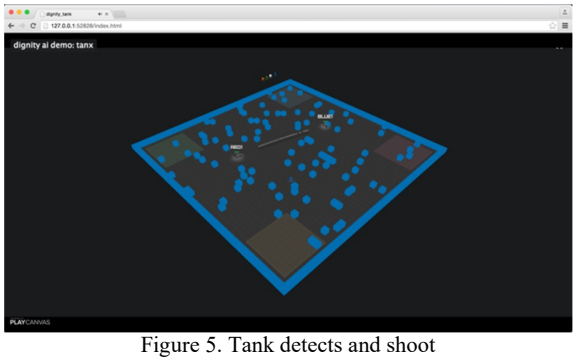

Figure 5. Tank detects and shoot

6. CONCLUSION AND RECOMMENDATIONS

While having an object, which can be used in HTML5 game engines, object oriented structure, the developed artificial intelligence library also contains the necessary items in a HTML5 based artificial intelligence library together with action management and navigation functions.
With a flexible structure, DignityMission class carries out the operation of missions, switching between missions and mission management.

While DignityAction class is able to manage the basic actions by means of move(), sense(), create(), destroy() and upgrade() methods, it can be customized with custom() method and this provides the flexible operation of the system by supporting the actions which can be defined apart from these five actions.

Among the existing methods on small and middle sized maps, with A* algorithm and three basic distance calculation methods, Manhattan and Diagonal and Euclidean, DignityPath class is the one that works simplest and most consistent. However, DignityPath class will be more powerful by adding other algorithms and also different calculation methods.

DignityAIBase class manages the basic artificial intelligence functions as well as being the basic class of the customized classes. With this aspect, it is in the backbone of the system. DignityVehicle, DignityHuman, DignityAnimal and Dignity Structure classes, which are inherited from DignityAIBase, get more powerful and more inclusionary when customized within themselves.

Also, by being open source, developing of DignityAI will move even faster by adapting it to other popular game engines. It will take its place in the literature by being a complete artificial intelligence library, which can carry out all the functions such as situation management, navigation, mission management and action management.

In this regard, first development was made on PlayCanvas game engine. Most important reason to choose PlayCanvas

game engine is because time to start developing game on this one is fairly short and its game engine infrastructure is quite powerful. Also with the help of PlayCanvas' online editor, stage design and 3D game design can be done easily. In addition to all these there are open source games and ready game models in PlayCanvas. These open source games can be forked from the main project as required and keep on developing at its own side. Thus, making it possible to spare more time to DignityAI's own dynamics and powerful structure rather than struggling with the game engine's and game's components.

In the next phase of DignityAI, the target is; to develop the system by adapting it to all artificial engine dynamics and HTML5 game methods and even by moving out of the HTML5 boundaries and contribute to HTML5's development and adapting it to Unity, Unreal Engine and CryEngine game engines, which are more global game engines. For this DignityAI is published as fully open source at social code sharing site Github [19]. Also, the sample application made with DignityAI is published through Github [20].

It is foreseen that the next model of the DigntyAI will be a model which can think, add its own missions by itself and choose its own actions and learn.

REFERENCES

[1] J. Wexler, «Artificial Intelligence in Games: A look at the smarts behind Lionhead Studio's "Black and White" and where it can and will go in the future,» 2002. [Çevrimiçi]. Available: http://www.cs.rochester.edu/~brown/242/assts/termprojs/games.pdf. [%1 tarihinde erişilmiştir19 April 2016].

[2] Z. Middleton, «Case History: The Evolution of Artificial Intelligence in Computer Games,» 2002. [Çevrimiçi]. Available: http://web.stanford.edu/group/htgg/sts145papers/zmiddleton_2002_1.pdf. [Accessed 19 04 2016].

[3] M. Stall, «My baby, the finite state machine,» 2006. [Çevrimiçi]. Available: http://blogs.msdn.com/b/jmstall/archive/2006/09/13/baby-state-machine.aspx. [Accessed 21 04 2016].

[4] B. Schwab, AI game engine programming, Boston, Massachusetts: Cengage Learning, 2009.

[5] P. Lubbers, B. Albers ve F. Salim, Pro HTML5 programming, New York: Apress, 2011.

[6] A. Freeman, The Definitive Guide to HTML5, New York: Apress, 2011.

[7] T. Parisi, WebGL: up and running, California: O'Reilly Media, 2012.

[8] S. Rabin, AI Game Programming Wisdom, Newton Centre: Charles River Media, 2002.

[9] K. Compton and M. Mateas, "Procedural Level Design for Platform Games," in *In Proceedings Of The Second Artificial Intelligence And Interactive Digital Entertainment Conference*, California, 2006.

[10] D. Beirne, "Racing Game AI: An Investigation into AI Techniques for Motorsport Simulation Games," 2007. [Online]. Available: http://www.mygamedemos.com/Abertay/David%20Beirne%20CS%201130A%20Artificial%20Intelligence%20for%20Games%20-%20Racing%20Game%20AI.pdf. [Accessed 21 04 2016].

[11] J. Gordon, «A finite state machine javascript micro framework. Github Repository,» 2011. [Çevrimiçi]. Available: https://github.com/jakesgordon/javascript-state-machine. [Accessed 21 04 2016].

[12] J. Cowart, «js ex machina - finite state machines in JavaScript. Github Repository,» 2012. [Çevrimiçi]. Available: https://github.com/ifandelse/machina.js. [Accessed 21 04 2016].

[13] X. Xu, «A comprehensive path-finding library for grid based games. Github Repository,» 2011. [Çevrimiçi]. Available: https://github.com/qiao/PathFinding.js. [Accessed 21 04 2016].

[14] A. Subagio, Learning Construct 2: Design and create your own engaging, extensible and addictive game using Construct 2, Birmingham: Packt Publishing, 2014.

[15] D. Cielen ve A. Meysman, HTML5 Game Development with ImpactJS: A step-by-step guide to developing your own 2D games, Birmingham: Packt Publishing, 2013.

[16] B. Bibat, "HTML 5 Shoot'em Up in an Afternoon: Learn (or teach) the basics of Game Programming with this free Phaser tutorial," 2014. [Online]. Available: https://leanpub.com/html5shootemupinanafternoon/read. [Accessed 21 04 2016].

[17] P. Kashyap, «Investigation into the use of HTML 5 game engines to create a responsive social educational game for children,» 2015. [Çevrimiçi]. Available: https://espace.cdu.edu.au/eserv/cdu:46185/Thesis_CDU_46185_Kashyap_P.pdf. [Accessed 21 04 2016].

[18] B. McInnis, R. Shimizu, H. Furukawa, R. Fushimi, R. Tanaka ve K. Kratzer, HTML5

Game Programming with enchant.js, New York: Apress, 2013.

[19] B. Uslu, «DignityAI - AI Library for HTML5 Games. Github Repository,» 2015. [Çevrimiçi]. Available: https://github.com/berkanuslu/dignityai. [Accessed 21 04 2016].

[20] B. Uslu, «DignityAI Demo 1: Tanx AI. Github Repository,» 2015. [Çevrimiçi]. Available: https://github.com/berkanuslu/dignity_tank. [Accessed 21 04 2016].