10-1-2015

A-Maze-D: Advanced Maze Development Kit Using Constraint Databases

Shruti Daggumarti
University of Nebraska-Lincoln, srd291@gmail.com

Peter Revesz
University of Nebraska-Lincoln, prevesz1@unl.edu

Corey Svehla
University of Nebraska-Lincoln, csvehla1236@huskers.unl.edu

A-Maze-D: Advanced Maze Development Kit Using Constraint Databases

Shruti Daggumati, Peter Z. Revesz, and Corey Svehla
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, Nebraska 68588-0115
Email: sdagguma@cse.unl.edu, revesz@cse.unl.edu, csvehla@unl.edu
http://cse.unl.edu/revesz Telephone: (1+) 402 472–3488

Abstract—In this paper, we describe the A-Maze-D system which shows that constraint databases can be applied conveniently and efficiently to the design of maze games. A-Maze-D provides a versatile set of features by a combination of a MATLAB library and the MLPQ constraint database system. A-Maze-D is the first system that uses constraint databases to build maze games and opens new ideas in video game development. *Keywords*—nimation, constraint database, maze, MLPQ, moving objects, video gamenimation, constraint database, maze, MLPQ, moving objects, video gamea

I. Introduction

The rapidly growing video game industry has a revenue of approximately twelve billion U.S. dollars per year in the United States alone (Statista [8]). The efficient development of new video game products is needed to supply the growing demands for video games that have become ubiquitous on our computers, consoles, and phones.

A large set of video games require the representation of a map, usually some kind of maze, and other spatial objects. In addition, video games also routinely require the representation of moving objects. Hence video games have a strong connection with geographic, spatial and moving object (also called spatio-temporal) databases. Since these types of databases can be viewed as special cases of constraint databases (Kanellakis et al. [2], Revesz [3]), we propose *A-Maze-D*, an *Advanced Maze Development* kit with a novel design based on the MLPQ system (Revesz et al. [4]). The MLPQ system, developed at the University of Nebraska-Lincoln, is one of the systems that implements and visualizes constraint databases and has been used already in many spatial and moving object database applications.

Our proposed A-Maze-D system provides a large set of useful features that enable game development where the main objective is to find the way out of a maze with limited viewing distance from an overhead view. We describe in detail the features that are the most important in developing maze games. We envision that with a growing set of features, the development can be extended from maze games to a plethora of different types of video games.

This paper is organized as follows. Section 2 describes some related work on mazes and constraint databases. Section 3 presents the A-Maze-D system. Section 4 provides some conclusions and future work.

II. Related Work

A. Mazes and Maze Games

A maze is a complex passage with multiple branches where the user needs to find the best route. In most mazes, the walls are fixed and do not change as the user progresses through the maze. Maze solving computer algorithms that try to find the best path to an exit or the fastest way to attain a prize have been implemented many times before. However, the typical solutions do not take into account a user's limited vision and inability to mark the walls as hindrances for solving the mazes. In Section 3, we specifically allow the option of representing the limited vision of a user.

In games like Super Mario World, there are levels where the user needs to solve a maze and has limitations as to what they can see. We see this type of maze in many different games including some Legend of Zelda games. In addition, in video games such as Super Mario World if the user has failed a level numerous times then the system offers the ability for the system to show the proper way to finish the level where obstacles of all kinds are taken into account and the shortest path is used.

In well-known games such as Pac-Man, the objective is to collect all the pellets and to live as long as the ghosts do not eat the user. In other games, the goal is to rescue a princess or to find the treasure and the end of the journey. We choose to favor the idea of there being some form of treasure at the end of the levels.

Mazes form also an important element in biology in the study of learning. Rodents were first used in mazes by Willard Small from Clark University (Small [5]). Using rodent burrows as a design a maze was created to test the cognitive abilities of rodents. Soon after these early experiments, different animals were used for testing purposes ranging from monkeys to birds (Watson [7]). James Watson also sent rodents through a maze with some sensory deprivations. Fleming Perrin of the University of Chicago tested humans where he blindfolded each person and let them solve a dodecagonal maze (Perrin [6]). However, rats remain in biology the primary test subject for mazes. Many of these types of experiments can be modeled in our maze choice game.

ISBN: 978-1-61804-350-4

B. Constraint Databases

Constraint databases (Kanellakis et al. [2], Revesz [3]) provide an extension of relational databases where the input data consists of constraint relations. Each constraint relation is a set of constraint tuples. In constraint tuples, the attributes are referred to by variables and the possible values of the attribute variables are restricted by constraints. In particular, the MLPQ constraint database system uses linear constraints on the attribute variables (Kanellakis et al. [2], Revesz [3]).

III. The A-Maze-D (Advanced Maze Developer) Kit

The A-Maze-D, short for Advanced Maze Developer, kit is a versatile system that enables efficient development of complex maze games. The A-Maze-D system consists of a MATLAB library that allows the easy translation of input data into MLPQ system input files. Once all stationary and moving objects are translated into MLPQ input files, the input files can be opened and animated in the MLPQ system. The animation lasts until some choice point is reached. The choice point requires that the user enter some input parameters, such as whether to turn left or right at the current location in the maze, to fire some bullets, to start a conversation or some other action. Once the input parameters are entered the corresponding animation can continue until the next choice point.

We organize the description of the A-Maze-D system into the following subsections. Section III-A shows how A-Maze-D can specify stationary objects such as mazes with either straight or curved walls. Section III-B describes the specification of moving objects such as persons, exploding objects and shields.

A. Stationary Objects

The main stationary object in a maze game is the maze itself. Figure 1 shows two mazes with walls that are composed of straight lines, while Figure 4 illustrates a maze with curved lines. The implementation of mazes, especially with curved walls, can be a tedious software engineering task. However, we show below that using the A-Maze-D system mazes can be developed efficiently whether the mazes have straight or curved walls.

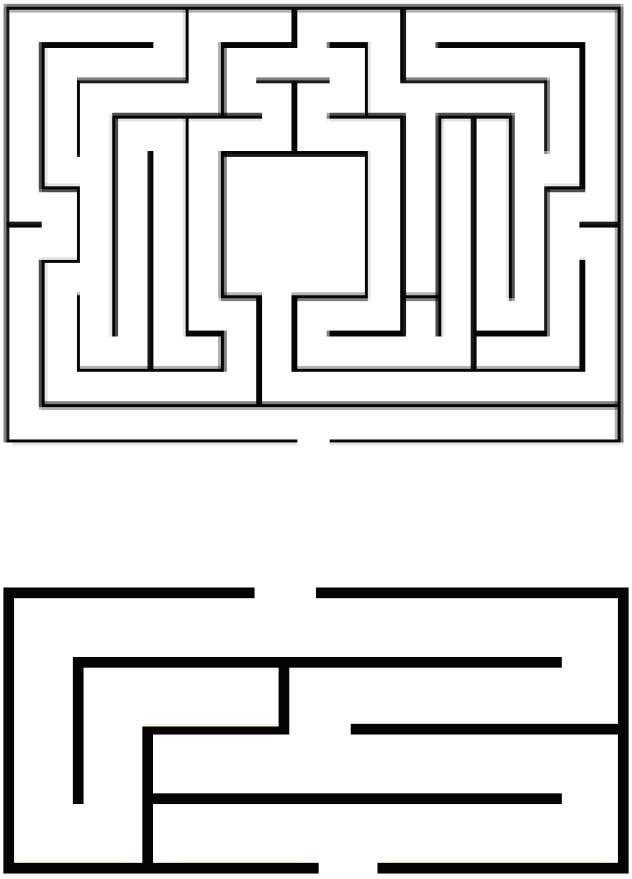

Fig. 1. Examples of mazes with straight walls.

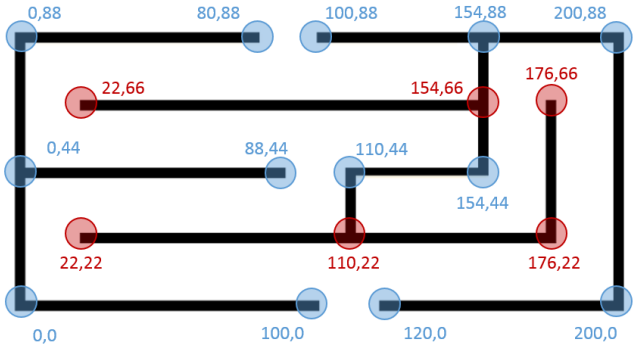

Fig. 2. Examples points on a maze.

Mazes with straight walls: Suppose that we would like to implement the maze shown on the left side of Figure 1. At first, we record the corner points for each wall as shown with highlights in Figure 2.

Each wall needs to have a different id. In the case of the maze in Figure 2 we need six walls with the (x, y) corner points shown in Table III-A.

A-Maze-D has a library of MATLAB scripts that contains a function called *buildWalls* that takes as input the set of points and turns them into an MLPQ input file that represents the walls. The basic idea behind the *buildWalls* function is illustrated in Figure 3, which shows on the left a simple table with two points and on the right their implied meaning. The *buildWalls* function also needs a parameter that specifies the width for the wall. In this case for simplicity we chose a width of one. As can be seen, the (a, c) and (b, d) points define a parallelogram with width one and whose lower boundary line is the line segment from (a, c) to (b, d).

The *buildWalls* function transforms the points described in Table III-A into the following MLPQ input file:

ISBN: 978-1-61804-350-4

TABLE I
INPUT: CORNER POINTS ON THE MAZE.

id	X	Y
1	100	0
1	0	0
1	0	88
1	80	88
2	120	0
2	200	0
2	200	88
2	100	88
3	0	44
3	88	44
4	22	22
4	176	22
4	176	66
5	110	22
5	110	44
5	154	44
5	154	88
6	22	66
6	154	66

X	Y
a	c
b	d

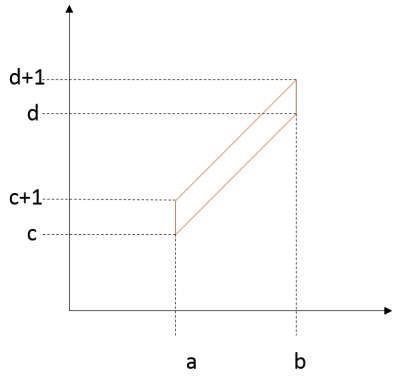

Fig. 3. The points from the table are represented visually on the right.

```
begin %MLPQ%
R(id,x,y) :- id=1, x>=0, x<=100, y=0.
R(id,x,y) :- id=1, x=0, y>=0, y<=88.
R(id,x,y) :- id=1, x>=0, x<=80, y=88.
R(id,x,y) :- id=2, x>=120, x<=200, y=0.
R(id,x,y) :- id=2, x=200, y>=0, y<=88.
R(id,x,y) :- id=2, x>=100, x<=200, y=88.
R(id,x,y) :- id=3, x>=0, x<=88, y=44.
R(id,x,y) :- id=4, x>=22, x<=176, y=22.
R(id,x,y) :- id=4, x=176, y>=22, y<=66.
R(id,x,y) :- id=5, x=110, y>=22, y<=44.
R(id,x,y) :- id=5, x>=110, x<=154, y=44.
R(id,x,y) :- id=5, x=154, y>=44, y<=88.
R(id,x,y) :- id=6, x>=22, x<=154, y=66.
end %MLPQ%
```

Mazes with curved walls: The A-Maze-D system uses multiple MATLAB scripts in order to attain the smoothest-looking curved walls. For example, Figure 4 shows a maze where all walls are curved except for six straight walls that are all horizontal and are used to block further passage in maze at dead ends.

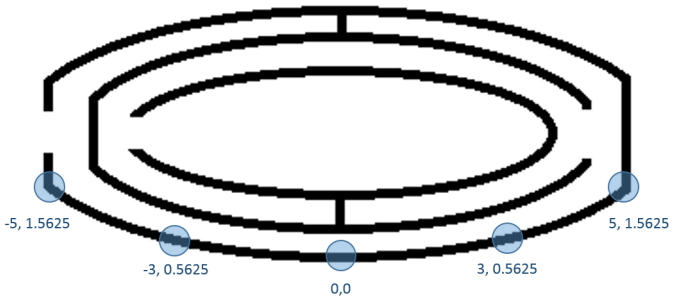

Fig. 4. Example of a curved maze.

Let us illustrate the A-Maze-D functions using the bottom curved wall, which is a segment of the parabola $y = (x/4)^2$. An approximation of the parabola can be specified by five points as shown in Figure 4. Our MATLAB script takes the x coordinates and the parabolic function $y = (x/4)^2$ to generate Table III-A. The script can take any other polynomial function. In case the boundary of the wall cannot be described by the user as a polynomial function, the A-Maze-D system also provides an alternative MATLAB script that makes a cubic spline interpolation for the given sample points of the wall.

TABLE II
EXAMPLE CORNER POINTS ON A MAZE.

id	X	Y
1	-5	1.5625
1	-3	0.5625
1	0	0
1	3	0.5625
1	5	1.5625

The more points we choose for the approximation, the smoother-looking parabolic curved wall we obtain. However, the smoother representation generates a larger MLPQ input file, which means generally a slower visualization and animations of the maze game. Given the above input data, the A-Mazed-D system already generates a large MLPQ output file, which begins as follows:

```
begin %MLPQ%

R(id,x,y) :- id=1, x>=-4.0000, x<=-3.9200,
             -0.640043x - y = 1.560172.

R(id,x,y) :- id=1, x>=-3.9200, x<=-3.8400,
             -0.588340x - y = 1.357495.

R(id,x,y) :- id=1, x>=-3.8400, x<=-3.7600,
             -0.541344x - y = 1.177033.
...

end %MLPQ%
```

ISBN: 978-1-61804-350-4

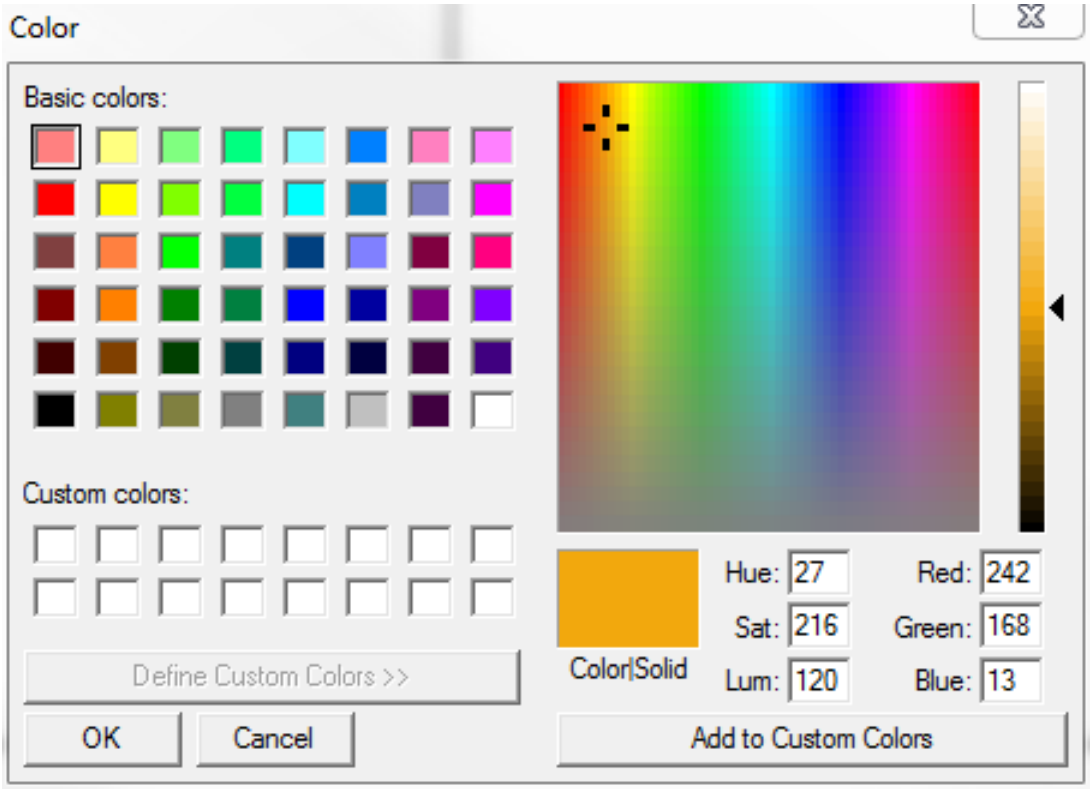

Fig. 6. Color Scheme Window

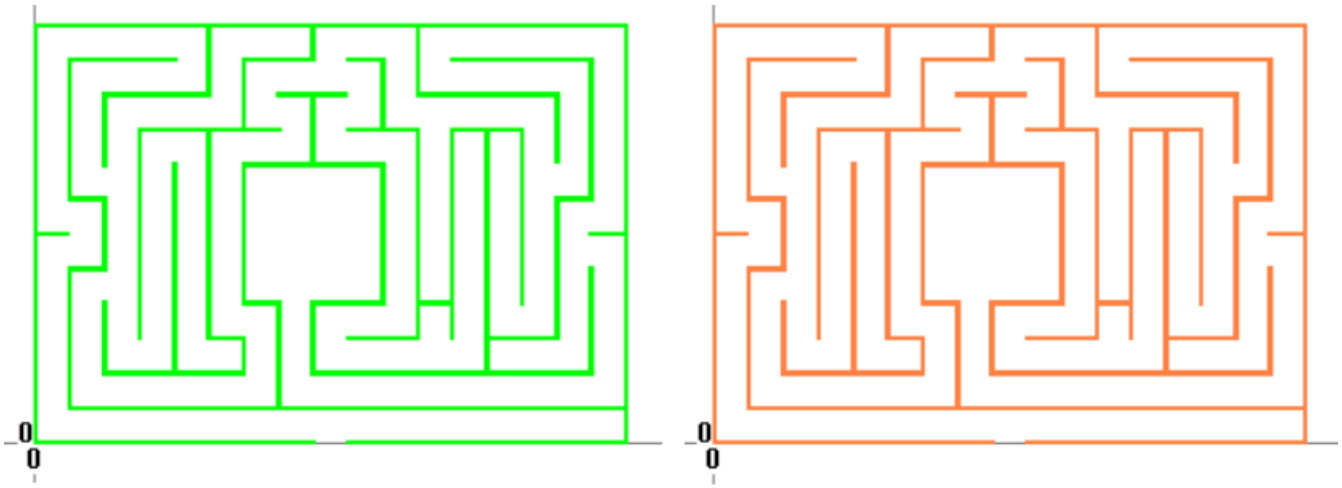

Fig. 7. Maze color changes from green (on the left)to orange (on the right)

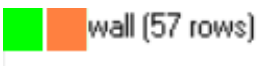

Fig. 5. Color Example

Color Change: We call *color animation* when stationary or moving objects change color over time. Color animation can provide important visual cues to the users. For example, in our maze we could make the walls change color that would show some type of time limit where the walls will start to change to a specific color when the user is running out of time. MLPQ handles color animation by allowing the user to choose two different colors for each relation.

When an MLPQ input file is loaded into the MLPQ system, then next to each spatial or moving object relation there are two colored boxes as shown in Figure 5. The first box represents the starting color, and the second box the ending color for the displayed relation during the animation. The MLPQ system provides a random pair of colors for each relation after a new file is uploaded. To change the colors to the desired values, we can double click on one of the boxes. Then a color scheme window will pop up as shown in Figure 6. This window allows the user to create the specific color of their

ISBN: 978-1-61804-350-4

choice for the selected box. For example, Figure 7 shows two snapshots of the color animation of a maze, which changes from green to orange.

B. Moving Objects

Player Movement: Movement of the player is guided by choices that the player can select in MLPQ. When the player chooses a movement option in the maze, the A-Maze-D system animates the player moving through the maze until the player reaches the next choice point where another decision is required. At any moment in time, each player is assumed to occupy a 4 by 4 square area. The movement of the square is represented by the attributes x, y and t, which denoted time. For a moving object, x and y are functions of time represented by linear constraints in the MLPQ system input files. The A-Maze-D system uses another MATLAB script to generate the MLPQ file.

The MATLAB script will automatically compile this and return the movements for your player to take. Once the constraints are created from the script you then have to put them in your MLPQ maze file with the proper relations. The code below is one example that shows how the constraints are stored inside the MLPQ file.

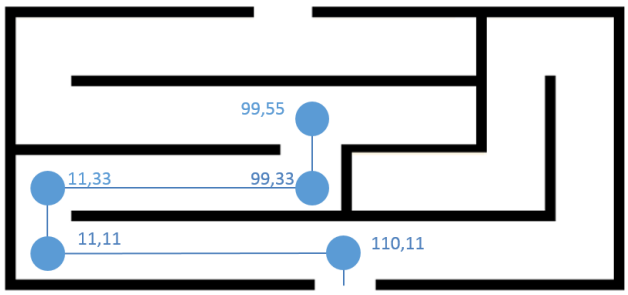

Fig. 8. A moving point within a maze.

For example, Figure 8 shows the movement of the player from location $(110, 110$ to location $(99, 55)$. Besides these beginning and ending points, we also record some of the other points that are on the way with the restriction that all turning points need to be included in the list. In this case, the selected points can be represented in the following table:

TABLE III
INPUT: CENTER POINTS FOR PLAYER MOVEMENT ON THE MAZE.

id	X	Y
1	110	11
1	11	11
1	11	33
1	99	33
1	99	55

The A-Maze-D system generates the following MLPQ file for the input shown in Table III-B.

```
begin %MLPQ%

player(id,x,y,t) :- id=1,
                    x + t >= 119,
                    x + t <= 121,
                    y > 10, y <= 11,
                    t >= 10, t <= 109.

player(id,x,y,t) :- id=1,
                    x >= 10, x <= 12,
                    y - t > -99,
                    y - t <= -98,
                    t >= 109, t <= 131.

player(id,x,y,t) :- id=1,
                    x - t >= -121,
                    x - t <= -119,
                    y > 32, y <= 33,
                    t >= 131, t <= 219.

player(id,x,y,t) :- id=1,
                    x >= 98, x <= 100,
                    y - t > -187,
                    y - t <= -186,
                    t >= 219, t <= 241.

end %MLPQ%
```

The A-Maze-D system generates an animation using MLPQ as a basis.

Searchlight: The A-Maze-D system also allows a limited field of view for the game player by adding a larger square relation around the moving player. The complement of that larger square will be displayed in black over the actual maze. For example, Figure 9 shows the limited field of view provided by a search light. The searchlight follows the player as he or she moves in the maze.

Explosions: The A-Maze-D system can represent explosions and other expanding objects. For fireworks can be represented as an object that expands until it ceases to exist. The A-Maze-D system provides a function that gives as input the beginning and the ending shapes of the exploding object and gives as output a parametric rectangles representation of the expanding object that is an accepted MLPQ input file.

Color Change: Color animation can be applied to moving objects too similarly as it is applied to stationary objects. For example, the firework could be black at the beginning and gradually lighted up and become completely red in the end.

ISBN: 978-1-61804-350-4

Fig. 9. Searchlight Example

IV. Conclusion and Future Work

Constraint databases and video games have been around for a long time, but they have never been combined together before. The A-Maze-D system shows that constraint databases can be applied conveniently and efficient to design of maze games. A-Maze-D provides a versatile set of features by a combination of a MATLAB library and the MLPQ constraint database system. This is the first time that maze games have been created using constraint databases.

In the future we would like to try implement different game types in constraint databases and see how well they can be converted over from the normal game development into constraint databases. We believe the best way to do this is testing out different types of games and see how well they perform. A few examples to name would be to test games that are more like Tetris where one needs a faster reaction or one's actions are limited by the time allotted.

Examining new possible features that constraint databases can provide for game development is another area we would like to research more and test out. Constraint databases have been used before in the animation of human faces. Hence an intriguing possibility is to allow players to speak to each other. Whenever a player wants to say something a pop-up window would open and show an animation of the player's face as he or she speaks.

References

[1] M. Lewis and J. Jacobson, Game engines, *Communications of the ACM*, 45.1, 27, 2002.

[2] P. C. Kanellakis, G. M. Kuper and P. Z. Revesz, Constraint query languages, *Journal of Computer and System Sciences*, 51 (1), pp. 26-52, 1995.

[3] P. Z. Revesz, *Introduction to Databases: From Biological to Spatio-Temporal*, New York, USA: Springer, 2010.

[4] P. Z. Revesz, R. Chen, P. Kanjamala, Y. Li, Y. Liu and Y. Wang, The MLPQ/GIS constraint database system, *ACM SIGMOD International Conference on Management of Data*, ACM Press, 2000.

[5] W. S. Small, Experimental study of the mental processes of the rat. II *The American Journal of Psychology*, pp. 206–239, 1901.

[6] F. A. C. Perrin, An experimental and introspective study of the human learning process in the maze. *Psychological Monographs: General and Applied*, 16.4, i-97, 1914.

[7] John B. Watson, Kinsthetic and organic sensations: Their role in the reactions of the white rat to the maze. *The Psychological Review: Monograph Supplements*, 8.2, 1907.

[8] "Monthly U.S. Video Game Industry Revenue 2015 — Statistic." Statista. Web. 13 Apr. 2015. http://www.statista.com/statistics/201093/revenue-of-the-us-video-game-industry/