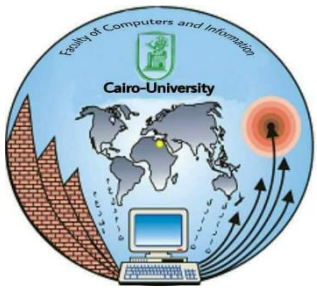
Cairo University

Faculty of Computers and Artificial Intelligence



Cairo University, Faculty of Computers and Artificial Intelligence

CS361 Artificial Intelligence

2nd Semester – 2020 - Project Description

Cairo University

Faculty of Computers and Artificial Intelligence

CS361

Artificial Intelligence

PID23867614

2nd Semester 2020 Project

Gamification

ID	Name	Email
20170022	Ahmed Sayed Mansour	Ahmed.mans20719@gmail.com
20170143	Abdelrahman Bahig	abdelrahmanbahig04@gmail.com
20170002	Ibrahim Ramadan Abdou	ibrahemramadan130@gmail.com
20170136	Atef Magdy Metwally	atefmagdy12@gmail.com
20170084	Hatem Sayed Ali	hatemgad98@gmail.com

a. Introduction (what is Gamification)

- **what is Gamification**

Gamification is about taking something that is not a game and incorporating game elements to improve consumer satisfaction, enjoyment and loyalty.

Gamification is to use game mechanics and designing experiences to engage and motivate people to achieve their goals digitally.

Gamification applies game elements into non-game settings to maximize interaction, such as a website, online group, learning management system or intranet business.

- **Game Mechanics**

Game mechanics are the rules and rewards, which appear on a digital platform within a program. Examples may include points, levels, missions, rankings, badges, and advances. Game mechanics is how the participants engage in a gamification program and receive next steps and feedback on achievements.

- **Game Dynamics**

Game dynamics refers to a set of emotions, behaviors and desires that resonate with people found in game mechanics. Highlights could involve competitiveness by leaderboards, teamwork through completing team tasks, group through having other players on a news page, selection through discovering different missions as they win special badges and surprises.

- **the Business Value of Gamification**

Gamification – at its heart – is about driving commitment to influence business outcomes. When people engage in your gamification initiative and participate, they

learn the best way to interact with your business, your products, your services and your brand. The consumer does not lose the market interest of gamification. Game mechanics engagement provides insightful data that can help to influence marketing campaigns, platform utilization and performance objectives. Every employee or customer interaction gives a better sense of where a participant is spending his or her time and what activities drive interest.

- **Benefits of gamification**

1. Increased user engagement is the main benefit of gamification. Beat the GMAT, for example, takes on a serious task and puts on that a fun spin. If the person has fun, then information is more likely to be retained. Gamification also makes for hands-on job implementations you are engaging in the learning process, instead of merely reading about an occurrence or topic. Real-time input is given to notify the customer whether he or she is performing well or not.

2. Competition is another gamification benefit. Many of us strive to be the best. You are motivated to outperform your competitors with the gamification. This makes you work harder to achieve optimal results. Another motivator is rewards and prizes Platforms for gamification can offer real world prizes for a job well done. Many people want the feeling of winning something, even though it is something seemingly tiny like a free cup of coffee.

- **Examples**

1. **The US Army**

   This defense institution uses some very well crafted, media-leaked training games to get more people interested in joining the US armed forces.

2. **Progress Bars**

   You have a progress bar running with this to reflect progress of your project. This can help people immensely, and inspire them to successfully complete the tasks.

3. **Nike and Run Club**

   Nike+ goods and services seek to expand the Nike experience outside purchase. The Nike + Run Club app is no exception to this law, and provides an incentive for all racing enthusiasts to enter the runner group through the application. This helps you to assess your progress and pursue a personalized training plan customized to your standard and objectives.

b. **Background** (How to apply game principles in non-game contexts)

Gamification strategies are designed to manipulate the innate impulses of individuals for socialization, learning, dominance, competitiveness, accomplishment, prestige, self-expression, altruism, or resolution, or merely their reaction to the presentation of a game or play scenario.

Gamification operates by inspiring players to indulge in positive activities, providing a route to dominance, and manipulating our natural psychological predisposition to participate in games.

Smart marketers to increase consumer engagement and influence consumer behavior use this. To achieve this, consumers should be rewarded for specific behavior with virtual items (such as points) (e.g., buying something, signing up, using the product, filling out their profile), and those virtual items should offer access to exclusive privileges and rewards, such as levels or prizes.

- **Gamification Principles**

You need to understand its values before you can start talking about using gamification to develop your company.

‘‘**At SCVNGR we like to joke that with seven game dynamics you can pretty much get anyone to do anything.**’’ -***Seth Priebatsch***-

The aim of the game dynamics is to predictably drive a user-wanted behavior. To do so we need to consider how people are acting. After all, as Gabe Zichermann, co-author of "Gamification by Design," states: "Gamification is psychology at 75% and technology at 25 per cent."

- **Fogg Behavior Model**

Stanford University developmental psychologist Prof. B.J. Fogg built a concept of behavioral transformation. It says that for a behavior to occur at the same time three factors will converge: **Motivation, Ability** and **Trigger**. The most critical part of this is that all three variables will combine simultaneously.

So, after all any gamified system must provide:

- Giving users motivation to do something (emotional investment, promise of reward, etc.).
- The ability to complete the action.
- Finally, a trigger or cue to complete the action.

The framework is all about timing – if not all elements shoot at the same moment, the player may lose confidence or get irritated. Below are few ideas of gamification you should integrate into your website and software.

- **Rewards**

  A Reward is something that you receive and something you feel good about. The significant element is the feeling good component. Consumers should be compensated with similar actions with virtual objects (e.g. points) (e.g., purchasing, filling out a form), and such virtual products will have access to special benefits and bonuses, such as tiers or incentives.

- **Loss Aversion**

  Many citizens generally tend to prevent damages over benefit creation. One way to move on with this is to give people everything they might risk straight away (unless they want to play). When you join Zynga's

Farmville, you get a starter farm. If you are not working the farm because you are taking care of the seeds, they can wither and die.

- **Status, competition and reputation**

  Inherently most citizen desire a better ranking, and not only to catch up, but to outdo the Joneses. That is why it is a smart thing for leaderboards. Having social achievements also encourages people to unify continuously, and remain motivated to achieve clear goals. Check out this lecture to read more on how to build credibility networks.

- **Feedback**

  Feedback informs users that they have recorded their planned behavior, and displays the effects of the operation. Seeing points collect as acts are performed provides a framework of simple and instant incentives. It is also an instant sign that the consumer is achieving their target. Continually accomplishing small goals in order to reach a larger goal is often what makes games addictive.

c. Development of the described Game

- **System component**

Function	Description
intialize()	This function to get a table with random numbers [-1:1].
printBoard(board)	A function to display a board with a specific format.
minThreeBoards(board, PrePosition)	A function to get a three boards with three vales {-1, 0, 1} to build the tree.
knowMove(pre,cur)	To know which direction we moved with the position of the previous and current index in board.
newPositions(board, curPosition)	To get all possible new index position in board.
move(board, curPosition, move)	Get a board with a new form by the move parameter passed.
newPos(curPosition, move)	Get the new position index on board by passing the move kind as parameter.
possibleMoves (board, curPosition)	Get a list that contains all possible boards with new positions.
alphabeta(board, curPosition, PrePosition, level, alpha, beta, maximizingPlayer, numberMoves, move) :	Alphabeta function consider as an essential function that returns the best move to the user that gives him the best path to win through using all previous functions. It contains both cases one for MAX turn and another one for MIN turn and calls itself by recursion ,also the base case to stop building the tree.

game(board, numberMoves)	The main function to play the game it calls the alphabeta function to get the best move then put random value at the previous position and so on until the player wins or number of moves finished.

- **Process of how code works**:
  - The main Function to be called is **Game** (board, numberMoves), it takes the board and number of moves allowed and it calls the **alphabeta** function to get the best move for the user to take. Therefore, the user can get better score through this move then the Game function replaces the previous position with a random number then call itself again until the player wins or the number of moves remain **none**.
- **Note:**
  - Function Initialize() is to get a board all of its numbers are randomly taken ,but I forced the beginning table to be like the example in the document.

- **Code Listing**

```
import random
import math

up    = [3 ,4 ,5 ,6 ,7 ,8]
down  = [0 ,1 ,2 ,3 ,4 ,5]
right = [0 ,1 ,3 ,4 ,6 ,7]
left  = [1 ,2 ,4 ,5 ,7 ,8]

def intialize():
    board = [0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0]
    for i in range(9):
        board[i] = random.randrange(-1, 2)
    return board

def printBoard(board):
    print(" "+ str(board[0])+ " | "+ str(board[1])+ " | "+
str(board[2]))
    print("_____________")
    print(" "+ str(board[3])+ " | "+ str(board[4])+ " | "+
str(board[5]))
    print("_____________")
    print(" "+ str(board[6])+ " | "+ str(board[7])+ " | "+
str(board[8]))


def minThreeBoards(board, PrePosition) :
    arr1, arr2, arr3 = list(board), list(board), list(board)
    arr1[PrePosition] = -1
    arr2[PrePosition] = 0
    arr3[PrePosition] = 1
    boards = [ arr1, arr2, arr3]
    return boards

def knowMove(pre,cur):
    if pre-cur == 3 :
        return "up"
    elif pre-cur == -3 :
        return "down"
    elif pre-cur == -1 :
        return "right"
    elif pre-cur == 1 :
        return "left"
```

```
def newPositions(board, curPosition):
    boards = []
    #UP
    if curPosition in up:
        boards.append(curPosition-3)
    #DOWN
    if curPosition in down:
        boards.append(curPosition+3)
    #RIGHT
    if curPosition in right:
        boards.append(curPosition+1)
    #LEFT
    if curPosition in left:
        boards.append(curPosition-1)
    return boards

def move(board, curPosition, move):
    if move == 'up':
        arr = list(board)
        arr[curPosition-3] = arr[curPosition-3] + arr[curPosition]
        return arr
    elif move == 'down':
        arr = list(board)
        arr[curPosition+3] = arr[curPosition+3] + arr[curPosition]
        return arr
    elif move == 'right':
        arr = list(board)
        arr[curPosition+1] = arr[curPosition+1] + arr[curPosition]
        return arr
    elif move == 'left':
        arr = list(board)
        arr[curPosition-1] = arr[curPosition-1] + arr[curPosition]
        return arr

def newPos(curPosition, move):
    if move == 'up':
        next = curPosition-3
        return next
    elif move == 'down':
        next = curPosition+3
        return next
    elif move == 'right':
        next = curPosition+1
        return next
    elif move == 'left':
        next = curPosition-1
        return next
```

```
def possibleMoves(board, curPosition):
    boards = []
    arr1, arr2, arr3, arr4 = list(board), list(board),
list(board), list(board)
    #UP
    if curPosition in up:
        arr1[curPosition-3] = arr1[curPosition-3] +
arr1[curPosition]
        boards.append(arr1)
    #DOWN
    if curPosition in down:
        arr2[curPosition+3] = arr2[curPosition+3] +
arr2[curPosition]
        boards.append(arr2)
    #RIGHT
    if curPosition in right:
        arr3[curPosition+1] = arr3[curPosition+1] +
arr3[curPosition]
        boards.append(arr3)
    #LEFT
    if curPosition in left:
        arr4[curPosition-1] = arr4[curPosition-1] +
arr4[curPosition]
        boards.append(arr4)
    return boards


def alphabeta(board, curPosition, PrePosition, level, alpha, beta,
maximizingPlayer, numberMoves, move) :
    returnList = []
    #TESTING
    #printBoard(board)
    #print(str(curPosition)+" "+ str(PrePosition)+" "+
str(level)+" "+ str(alpha)+" "+ str(beta)+" "+
str(maximizingPlayer))

    if  level >= numberMoves :
        returnList = [str(board[curPosition]) , move]
        return returnList
    if board[curPosition] >= goal :
        returnList = [str(board[curPosition]+10) , move]
        return returnList

    if maximizingPlayer :
        value = -1*math.inf
        new = list(newPositions(board, curPosition))
```

```
        boards = list(possibleMoves(board, curPosition))
        #print("MAX")
        bestMove = []
        #print(boards)
        for i in range(len(boards)):
            temp = value
            move.append(knowMove(curPosition, new[i]))
            #print(move)
            newVal = alphabeta(boards[i], new[i], curPosition,
level+1, alpha, beta, False, numberMoves, move)
            tempMove = list(move)
            move.pop()
            #print(tempMove)
            #print(level)
            #print(newVal)
            value = max(value, int(newVal[0]))
            if temp != value :
                bestMove = list(tempMove)

            alpha = max(alpha, value)
            if alpha >= beta :
                break

        #return value
        returnList = [str(value) , bestMove]
        return returnList
    else :
        value = math.inf
        boards = list(minThreeBoards(board, PrePosition))
        #print("MIN")
        #print(boards)
        for i in range(3):
            newVal = alphabeta(boards[i], curPosition,
PrePosition, level, alpha, beta, True, numberMoves, move)
            value = min(value, int(newVal[0]))
            beta = min(beta, value)
            if alpha >= beta :
                break
        #return value
        returnList = [str(value) , move]
        return returnList

def game(board, numberMoves):
    print("\nYour board :")
    printBoard(board)
    pre = -1
    position = 6
```

```
    temp = numberMoves
    for i in range(temp):
        if board[position] >= goal :
            print("\nWINNING in " + str(i+1) + " moves with score
of " + str(board[position]))
            print("Input :-\n   Number of moves : " + str(moves) +
"\n   Goal : " + str(goal) + "\n")
            break
        arr =  alphabeta(board, position, pre, 0, -1*math.inf,
math.inf, True, numberMoves, [])
        #print(arr)
        print("\nbest move for this state to take " + arr[1][0] +
" in move number " + str(i+1))
        newBoard = move(board, position,arr[1][0])
        board = newBoard
        pre = position
        position = newPos(position,arr[1][0])
        newBoard[pre] = random.randrange(-1, 2)
        printBoard(newBoard)
        numberMoves = numberMoves-1
        if i == temp-1 :
            if board[position] >= goal :
                print("\nWINNING in " + str(i+1) + " moves with
score of " + str(board[position]) + "\n")
                print("Input :-\n   Number of moves : " +
str(moves) + "\n   Goal : " + str(goal) + "\n")
            else :
                print("\nLOSING with score of " +
str(board[position]) + "\n")
                print("Input :-\n   Number of moves : " +
str(moves) + "\n   Goal : " + str(goal) + "\n")
            break

if __name__ == "__main__":
    goal = int(input("Goal = "))
    moves = int(input("Maximum number of moves = "))
    game([1,-1,0,1,0,1,0,1,-1],moves)
    #game([-1,1,1,-1,-1,1,1,1,1],moves)
    '''
    if you want to make the all the board values initialized with
random values call 'intialize()' function:
    EX:
    game(intialize(),moves)
    '''
```

- **Test Cases**

1. **Test Number one**

```
Goal = 2
Maximum number of moves = 4

Your board :
 1 | -1 | 0
_____________
 1 | 0 | 1
_____________
 0 | 1 | -1

best move for this state to take up in move number 1
 1 | -1 | 0
_____________
 1 | 0 | 1
_____________
 1 | 1 | -1

best move for this state to take up in move number 2
 2 | -1 | 0
_____________
 -1 | 0 | 1
_____________
 1 | 1 | -1

WINNING in 3 moves with score of 2
Input :-
   Number of moves : 4
   Goal : 2
```

2. Test Number two

Goal = 6
Maximum number of moves = 11

Your board :
1 | -1 | 0
_____
1 | 0 | 1
_____
0 | 1 | -1

best move for this state to take up in move number 1
1 | -1 | 0
_____
1 | 0 | 1
_____
0 | 1 | -1

best move for this state to take up in move number 2
2 | -1 | 0
_____
0 | 0 | 1
_____
0 | 1 | -1

best move for this state to take down in move number 3
1 | -1 | 0
_____
2 | 0 | 1
_____
0 | 1 | -1

best move for this state to take up in move number 4
3 | -1 | 0
_____
1 | 0 | 1
_____
0 | 1 | -1

best move for this state to take down in move number 5

1 | -1 | 0

_____

4 | 0 | 1

_____

0 | 1 | -1

best move for this state to take down in move number 6

1 | -1 | 0

_____

1 | 0 | 1

_____

4 | 1 | -1

best move for this state to take up in move number 7

1 | -1 | 0

_____

5 | 0 | 1

_____

-1 | 1 | -1

best move for this state to take up in move number 8

6 | -1 | 0

_____

0 | 0 | 1

_____

-1 | 1 | -1

WINNING in 9 moves with score of 6

Input :-

Number of moves : 11

Goal : 6

3. Test Number three

Goal = 10
Maximum number of moves = 15

Your board :
1 | -1 | 0
_____
1 | 0 | 1
_____
0 | 1 | -1

best move for this state to take up in move number 1
1 | -1 | 0
_____
1 | 0 | 1
_____
1 | 1 | -1

best move for this state to take up in move number 2
2 | -1 | 0
_____
0 | 0 | 1
_____
1 | 1 | -1

best move for this state to take down in move number 3
-1 | -1 | 0
_____
2 | 0 | 1
_____
1 | 1 | -1

best move for this state to take up in move number 4
1 | -1 | 0
_____
-1 | 0 | 1
_____
1 | 1 | -1

best move for this state to take down in move number 5

1 | -1 | 0

_____

0 | 0 | 1

_____

1 | 1 | -1

best move for this state to take up in move number 6

1 | -1 | 0

_____

1 | 0 | 1

_____

1 | 1 | -1

best move for this state to take down in move number 7

0 | -1 | 0

_____

2 | 0 | 1

_____

1 | 1 | -1

best move for this state to take up in move number 8

2 | -1 | 0

_____

1 | 0 | 1

_____

1 | 1 | -1

best move for this state to take down in move number 9

1 | -1 | 0

_____

3 | 0 | 1

_____

1 | 1 | -1

best move for this state to take up in move number 10

4 | -1 | 0

_____

1 | 0 | 1

_____

1 | 1 | -1

best move for this state to take down in move number 11

0 | -1 | 0

_____

5 | 0 | 1

_____

1 | 1 | -1

best move for this state to take down in move number 12

0 | -1 | 0

_____

1 | 0 | 1

_____

6 | 1 | -1

best move for this state to take up in move number 13

0 | -1 | 0

_____

7 | 0 | 1

_____

1 | 1 | -1

best move for this state to take down in move number 14

0 | -1 | 0

_____

0 | 0 | 1

_____

8 | 1 | -1

best move for this state to take right in move number 15

0 | -1 | 0

_____

0 | 0 | 1

_____

1 | 9 | -1

LOSING with score of 9

Input :-

Number of moves : 15

Goal : 10

4. Test Number four

Goal = 4
Maximum number of moves = 9

Your board :
1 | -1 | 0
_____
1 | 0 | 1
_____
0 | 1 | -1

best move for this state to take up in move number 1
1 | -1 | 0
_____
1 | 0 | 1
_____
-1 | 1 | -1

best move for this state to take up in move number 2
2 | -1 | 0
_____
0 | 0 | 1
_____
-1 | 1 | -1

best move for this state to take down in move number 3
0 | -1 | 0
_____
2 | 0 | 1
_____
-1 | 1 | -1

best move for this state to take up in move number 4

```
 2 | -1 | 0
_____________
 1 | 0 | 1
_____________
-1 | 1 | -1
```

best move for this state to take down in move number 5

```
-1 | -1 | 0
_____________
 3 | 0 | 1
_____________
-1 | 1 | -1
```

best move for this state to take right in move number 6

```
-1 | -1 | 0
_____________
-1 | 3 | 1
_____________
-1 | 1 | -1
```

best move for this state to take down in move number 7

```
-1 | -1 | 0
_____________
-1 | -1 | 1
_____________
-1 | 4 | -1
```

WINNING in 8 moves with score of 4

Input :-

Number of moves : 9

Goal : 4

d. References

- https://www.bunchball.com/gamification
- https://potion.social/en/blog/10-amazingly-successful-examples-of-gamification
- Robson, K., Plangger, K., Kietzmann, J., McCarthy, I. & Pitt, L. (2015). "Is it all a game? Understanding the principles of gamification".
- https://www.growthengineering.co.uk/definition-of-gamification/
- Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From Game Design Elements to Gamefulness: Defining "Gamification". Paper presented at the 15th International Academic MindTrek Conference, Tampere.
- Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall
- https://www.javatpoint.com/ai-alpha-beta-pruning
- https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/
- https://neilpatel.com/blog/gamification-for-better-results/