

The Mathematics Conceptual Game

Nunnapad Toadithep

nisachol@cp.su.ac.th

Faculty of Science, Silpakorn University, Nakhon Pathom 73000
Thailand

Abstract: Some suitable or appropriate games can make children improve their thinking in a numerical way or in a mathematical way. This may be the way to create the Great Mathematician. This paper explores conceptual thinking in playing a Pick1-2-3 game. This game needs two players to take turn in picking either 1, 2, or 3 balls in their turn. In this game, the last player who picks the last ball will lose the game. One thing to remember is not to leave 4 balls or less (except 1) to the opponent. To have a chance of winning the game, a player should leave the multiple of 4 plus one balls, for example 5, 9, 13, 17, 21,...etc.

1.Learning while playing

It is a fact that an infant can learn whilst she is in her mother womb. An infant can participate with her mother in happiness and sadness. And it is also a fact that a child's environment can stimulate their concentration, such as using colorful toys, polite words, and cheerful games. This is because little children like to play games rather than to study for an extended amount of time. They are happy to play, so it is a good thing to make children be in the situation of learning while playing.

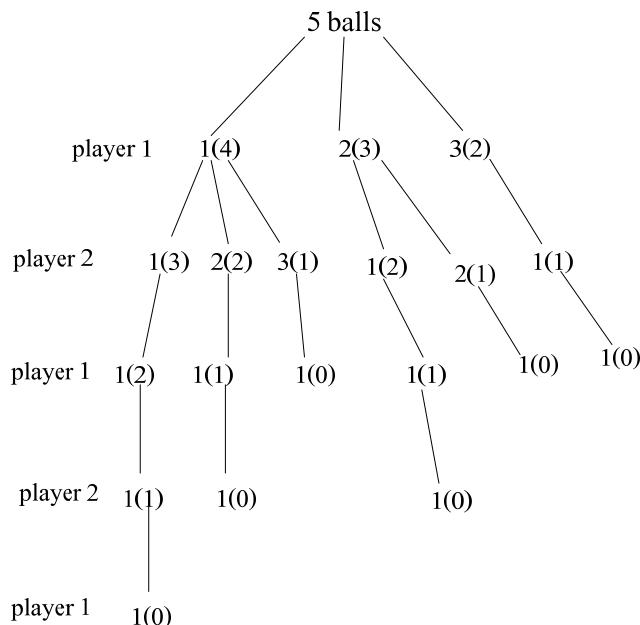


Figure 1: Player 1 loses the game in 4 of 6 possible ways

2. The Playing of a Pick1-2-3 Game

The principle rule of a Pick1-2-3 game is, two players take turns to pick either 1, or 2, or 3 balls. The player who picks the last ball will lose the game. The beginning of game is to randomly choose the number of balls to play. The tree diagram in Figure 1 shows all the possible ways of the two players to play the game of 5 balls. The first level of the tree is a beginning state, having 5 balls. The next level shows the possible ways of player 1 to play, the next level for the player 2 to play, and so forth. The notation 1(4) means pick 1 and leave 4 balls. By assuming that the 2 players are innocent in picking for the winner, but this thing does not happen in real life. If there are 4 balls left, the next player should be clever enough to pick 3 balls so forcing the opponent to take the last ball. Including the smart picking of the players, the tree diagram of the game should be as shown in Figure 2. It is clearly that with beginning of 5 balls, player 1 will lose the game in all possible ways of picking the balls.

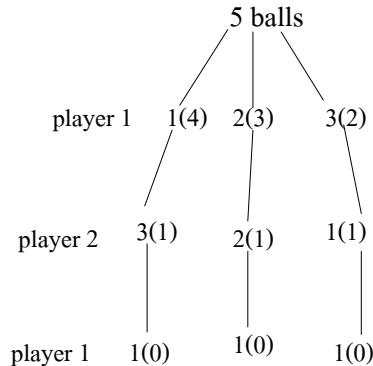


Figure 2: Picking of initial state of 5 balls

How will the game proceed if the number of balls at the initial state is 6?. The tree diagram in Figure 3 shows that if the player 1 picks 1 ball in the first turn, the player 1 can definitely win the game.

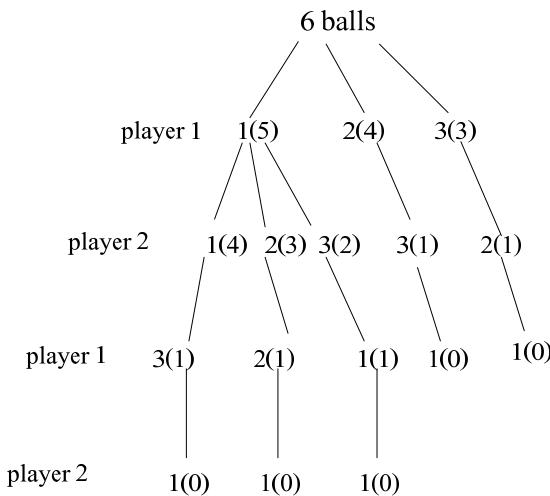


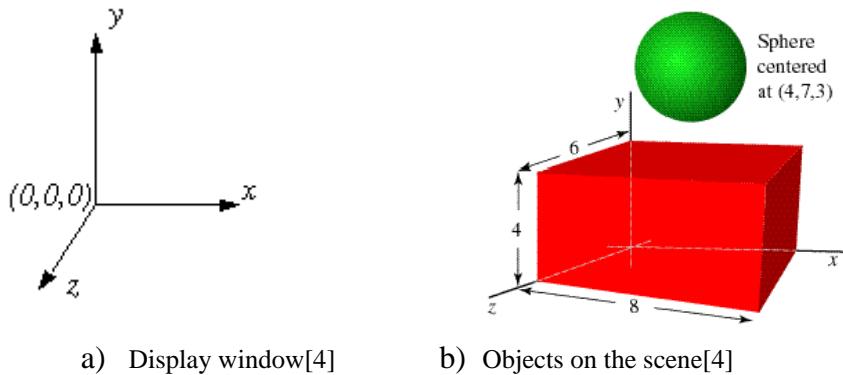
Figure 3: Picking 1 out of 6 balls leads to win the game

3. VPython and the Component of Games

3.1 Visual Objects

VPython is a Visual, 3D, and animated view of seeing. There are many 3D-objects in VPython. VPython gives the flexibility of object manipulating. With Object-Oriented Programming: objects have attributes or properties and methods to act with. There are 14 basic display objects listed below: cylinder, arrow, cone, pyramid, sphere, ring, box, ellipsoid, curve, helix, convex, label, frame and faces. To invoke the Visual module, place the following statement at the start of the file (program file, .py):

```
from visual import *
```



a) Display window[4] b) Objects on the scene[4]

Figure 4: VPython window and Objects

When using VPython the display window shows objects in 3D the origin (0,0,0) is in the center of the display window. The +x axis runs to the right, the +y axis runs up, and the +z axis points out of the screen, toward you, shown in Figure 4 a).

The graphical objects you create, such as spheres, boxes, and curves, continue to exist for the duration of your program, and the Visual 3D graphics module will continue to display them, wherever they are as in Figure 4 b). If you name the object, you can refer to that object by name such as RedBox, GreenBall. All objects have attributes or properties: there are built-in sets of attributes such as pos, color, radius. If you change an attribute of an object, such as its position or color, Visual will automatically display the object in its new location, or with its new color. For example the following command sets the position of GreenBall to (0,5,0) on the y-axis at scale 5 from the origin.

```
GreenBall.pos = (0,5,0)
```

If you want to create a ball, you should create it using a sphere object and if you want the balls to be in line with each other, you need to use the radius property like this: ball1, ball2 have radius 0.5, so ball2 should be set next to ball1, ball1.pos must be referred to and be add with

$2 * \text{radius}$ and set to be the pos property of ball2, in the x-axis , the first value of tuple : with these two VPython commands.

```
ball1 = sphere(pos=(1,2,1), radius=0.5)
ball2 = sphere(pos=(ball1.pos + (2*0.5,0,0) ), radius=0.5)
```

3.2 Random color and number of balls

For flexibility of game, this game needs the randomisation of 2 things: ball color and number of initial balls. VPython has 2 types of functions for randomisation. First function random() returns the floating point or real value from 0 to less than 1. This value is appropriate to the color tuple, (r,g,b). The color of ball1 is set to the function random(), with this command.

```
ball1 = sphere(pos =(1,2,1), color = (random(), random(), random()))
```

The number of balls must be an integer which can be obtained using the function randrange(beg, end) , which returns a number in the specified range, beg, beg +1, ... end-1.

3.3 User interface of games

This program allows the players using a mouse to pick a ball. The players cannot pick more than 3 balls in their turn and the picked balls will disappear from screen. The game continues until reaching the player who loses the game. Figures 5 to 8 show the flow of playing with the player 1 picking 1 ball, the player 2 picking 2 balls, and then the player 1 picking 2 balls which leaves 1 ball left, forcing the player 2 to lose the game.

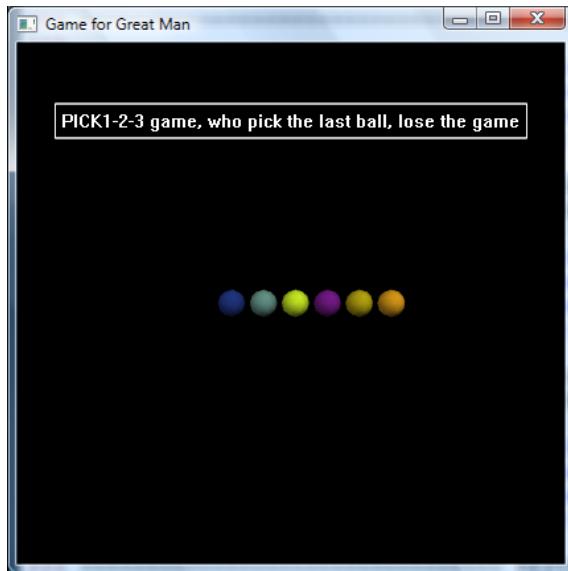


Figure 5: The initial state of 6 balls

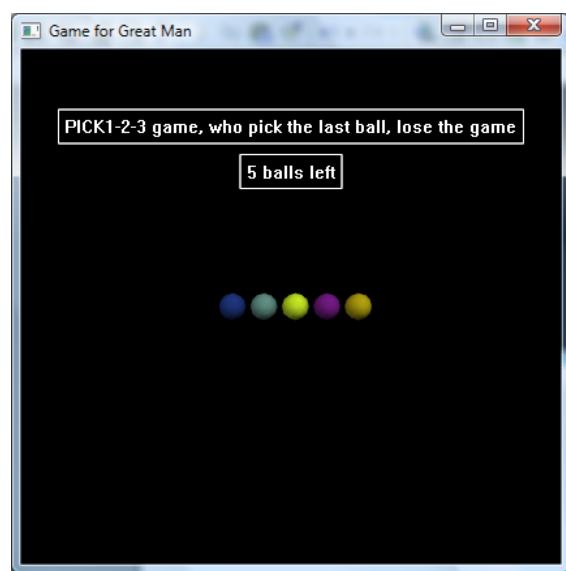


Figure 6: After the player 1 picks 1 ball

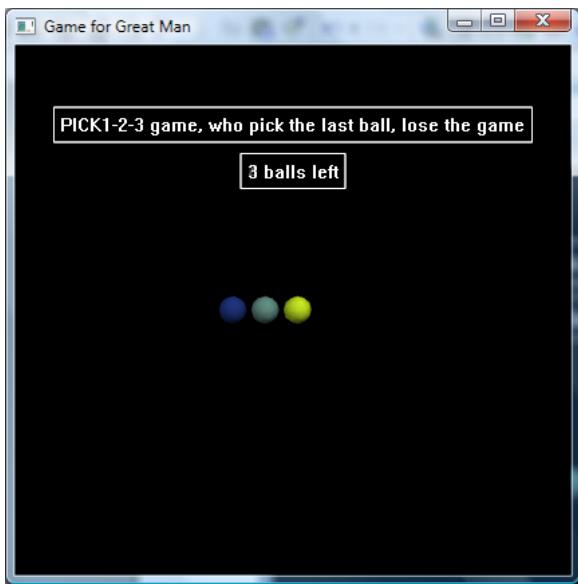


Figure 7: After the player 2 picks 2 balls

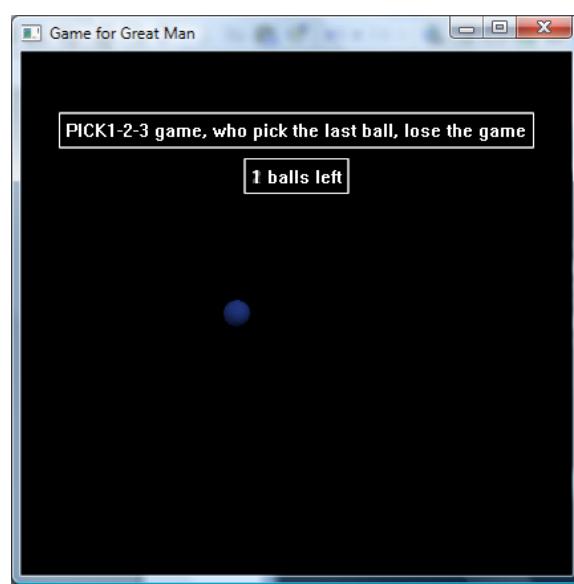


Figure 8: After the player 1 picks 2 balls

3.4 Three main functions in program

Here I will describe the three main functions in program. The first function is **Gen_balls(x,y)** that returns the number of generated balls and L, a object of list type in Python. There are two variables returned from this function which are n and L: n is a number of balls in the initial state, L is a list of n balls in different color placing in line on screen. The second function is **play(n,L)**, this function does the main task of game, allows the players to select the balls and shows the status of the number of the balls left. The last function, **randrange(5, 12)** is the function from module random in Python which returns the random number in [5..11] , in the above example this produced 6 balls.

```
def Gen_balls(x,y):
    n=randrange(5, 12) # random number from 5..11
    L=[]
    for i in range(n):
        rate(50)
        x=x+.7
        col1 = random()
        col2 = random()
        col3 = random()
        b=sphere (pos =(x,y,0), radius =.3, color=(col1,col2,col3))
        L.append (b)
    return n,L
```

```
def play(n, L):
    while 1:
        mm = scene.mouse.getevent()
```

```

while 1:
    mm = scene.mouse.getevent()
    if mm.pick in L :
        break
    # break from above while loop
    mm.pick.visible = 0
    n= n-1
    label(pos = (0, 3, 0), text="%d balls left" %n)

```

4. Conclusion

The rules of playing a Pick1-2-3 game are not so complicated. But if you wish to be a winner, you will have a trick to play. You will learn most from tree diagram in Figure 3: you should pick 1 ball and leave 5 balls to another player. Then you can definitely win this game, because the one who starts playing with 5 balls, can definitely lose the game. In any state, you should pick $m-1$ balls where $m= n \% 4$. Here n is the number of balls, m is the number of the remainder of the division of 4. For example if we have $n=11$, the remainder of division of 4 is 3 so that you should pick 2 balls. If you always keep on this trick of playing, you have more chance to win this game.

References

- [1] Binmore, K. (2005c). “*Fun and Games*”, second edition. Oxford: Oxford University Press.
- [2] Fudenberg, D., and Tirole, J. (1991). “*Game Theory*”. Cambridge, MA: MIT Press
- [3] Don Ross. “*Game Theory*”. Available from: <http://plato.stanford.edu/entries/game-theory/>
- [4] “The Visual Module of VPython”. <http://www.vpython.org/webdoc/index.html>