Dear all,

Greetings to you all and many thanks for giving me the chance to participate in this exciting contest! I will present you in some words the history of the game <Penalty kicks!>.

When i was young (14-17 years old, 20 years ago) i had tried to make this game in my amstrad 6128. I somewhat did it, but disk was lost, and there were no copies. So, now that i decided to make this game again, i had to do it all from the beginning. I remembered the general way i used to make the game, but nothing specific. In addition, i had not used amstrad for 20 years as drive belt was destroyed. Before 1 month i fixed it, and that was the fact that made me deal again with amstrad, but now in emulator mode mostly, as there is not enough room for the old one. I have to note that I don’t know anything about computer programming, and when i was young i knew few basic commands that i could understand from manual. Hopefully i found both manuals in pdf (greek and english) and that helped me remember many things. I think that what I made now is far much better from that game I had made in the past.

The commands that are used for this game are very few due to my limited knowledge. That’s why list of game is long, as I had to use many times the same commands in a very simple way in order to design or programme something. Commands used are mostly If/then, Print, Goto, Gosub/return, For/next, Input, Inkey$, move/plot/draw, locate, and chr$.

In order to design the goalpost and everything in the picture that does not move, i used move/draw commands. For designing the goalkeeper i used chr$ mostly (Basic graphics character set), but this didn’t work for all, so i had to add manually using move/draw some other features (legs). I did the same for designing the shooter. Pls note that when goalkeeper moves, i have first to disappear the <old> golakeeper and design the <new> in the appropriate position. In disappearing the keeper, I used print” “ (print blank), but this lead to re-designing the central parts of goalpost as they were too disappeared (nets in the centre) due to the blank print.

When the game starts, we have 3 possibilities: if the kicker shoots in the centre then K=0, if shoots in the left (as you watch the screen) K=1, and if he shoots in the right K=2. Using Goto we go in the appropriate line for each choice in order to specify then the other important variable, G. These 3 cases work (basically) in the same way.

Eg K=1.

Moving: to move the ball, we use print" " and print chr$(231) and of course the command locate. When ball disappears and appears to the next (diagonally in the case of K=1) point, it appers like it moves. In this stage we move the ball until it goes in front of the keepers line. But movement is too fast. How could we slow it? The answer is by using for/next. In EACH moving in this first stage, we add <for delay=1 to 31: next delay>. Amstrad needs time to calculate this, so moving now is slower and problem is solved. Now, during ball movement, we ask the player that handles the keeper to make his choice using inkey$. He chooses left or right by pressing the appropriate key, if nothing pressed then he stays in centre. And in this way we get the other variable G: keeper in centre G=0, keeper in left G=1, keeper in right G=2.

The use of inkey$ caused the following problem (regarding 2 player mode): if the kicker presses his button two times, then, in the specific case that the second kicker's press is earlier than the keeper's pressing button, the game recognizes inkey$ as the second press of the kicker, and due to that the keeper stays in the centre. This could be an easy cheat from the kicker that we had to correct. Also, this is easy to happen even if players are playing without wanting to cheat. We found the following solution: First of all, we can't send back the list for waiting the correct button using GOTO, as the ball is moving and we cannot interrupt this. So, we added a second inkey$ search, and now we have two inkey$ searches during ball movement with a small time difference. That way, the keeper's choice is received correctly. Of course we had to add again the if/then commands that define G depending on inkey$.

So far: the kicker has made the choice where to shoot, ball is moving in the specified direction until goalpost line (just before) and keeper is making the choice where to go. K and G are specified. K can range from 0 to 2, G can range from 0 to 2 too. So, we have to create 9 possible endings that include all possible combinations for K and G. In each ending we have to design the move of keeper (if moved), and the new ball movement. Pls note that we have to disappear keeper in case of moving and add central parts of goalpost that also disappeared. Each scenario ends with determining the variable GOAL (1 if we have goal, 0 if it is saved).

Problem: When game starts, the second teams shoots. So we have to change what inkey$ expects, as player 2 keys now decide the shoot direction (K). Another problem was how to count each team goals (as for every penalty we go back close to the line that the game begins). Another problem was how the game will decide when game ends.

Solutions to these came up using variable AKOL. When game starts, we define that AKOL=AKOL + 1. So, at first penalty of TEAM A, AKOL=1. Then, after first penalty of TEAM A is completed, game goes back for the first penalty of TEAM B- but now AKOL=2. So, that way, we know which penalty is in action (pls note that <Penalty No> that is shown in the game is not AKOL. If AKOL=2 then Penalty No=1 as it is the first penalty of TEAM B. Of course, we used AKOL to define the Penalty No Variable-PNO).

Using AKOL we can define variable PLIK. Variable PLIK is useful because it clarifies which team is shooting. It helps us because we don't have to write again and again all AKOL possible numbers (until 22). So, if AKOL= 1 or 3 or 5, or 7.....21) then PLIK=1 (Team A shoots) and in the other cases PLIK=2.

So, using now PLIK we first use it to define which keys are used for inkey$ in order to define K (kicker's choice). Then, we go at ball moving (each case for K=0 or 1 or 2) and we define which keys are used for inkey$ (depending on PLIK) to define keeper's choice. In this way, we have solved the controls problem. PLIK also helps us to add GOAL in the correct team. If PLIK=1 then goal is added in team A (SCORA=SCORA+GOAL), if PLIK=2 it is added in TEAM B (SCORB=SCORB+GOAL). In this way, if penalty is saved, GOAL=0, so nothing is added in the specified team score.

For game ending, we thought that we could use following:

When AKOL>10 (so after 5 shoots for each team) then ... (examine whose score is greater, SCORA OR SCORB)

But that was wrong! Note the case that we have a draw after 10 Penalties taken. When Team A scores the first penalty of <sudden death> (6th penalty of the team), AKOL>10 (11 in this case), SCORA>SCORB, but Team B hasn't shot yet! And game finishes!

So, the solution is that we have to manually enter all options, starting from When AKOL=10, when AKOL=12, when AKOL=14 ... until AKOL=22, of course without comparing scores when AKOL=11 or 13 etc).

This is the reason for the fact that the game ends with draw when 22 penalties are taken and we have no winner. If we could use AKOL>10, game could go as far as we could. But we don't think that this is serious problem that affects game, as most games would not go that far.

Another problem is that i am used to programme in Amstrad 6128 and the contest is for CPC 464. This has to do with some commands of 6128 that do not exist in CPC 464. Hopefully, i generally don't use many, and i avoided using them in the game. These commands are <clear input> and Fill.

Clear input was not used -when we needed this command, we added another variable$ for inkey$ that was not used anymore. Also, by using the option <press space to continue> all other keys except space will be ignored. As for Fill, we planned at the beginning to use it for making the goalpost bar more thick. But we did it appearing more thick, by drawing the same lines in the next pixels (move/draw commands) - you can see that goalpost bars is thicker that other lines, but this happens because each goalpost bar consists of 3 lines very near to each other.

Another problem we have to mention is about the position of penalty. The problem was that due to the fact that we use characters for moving the ball, in mode 1, the point of the initial position of the ball was too front (very near to the goalkeeper). We have switched to mode 2 and it seems better, but not perfect, as it is a bit more in front that it should. But we could not do anything more for this, from the moment that we decided that movement is depended on characters and not pixels and this movement is diagonal and based on characters.

Another problem was the speed of the moving ball. Speed of moving, as said previously, is fixed by added delay in each move using for/next command. But how much fast it must be moving? This game is a penalty game. So, as in penalties, keeper must not be able to see where the ball goes and then have the time to react correctly in most cases. This can happen only in rear cases and with full concentration and very good reflexes. So, after much practice, we decided that the correct point is 31. (for delay=1 to 31; next delay). In 9 possible endings (so, after keeper reaction has been specified) we used delay=1 to 40. This affects movement only in the last line in front of the goalpost and movement after the save (we can suppose that ball is also in reality a bit more slow or fast after the save. Also, difference is small and it doesn't seem to affect watching ball moving smoothly). In sum, when I wanted to change (increase or decrease) the reaction time of the goalkeeper, I had to go at ALL K cases (K=0, K=1, K=2) but NOT to the 9 possible endings (as when we get there, keeper has already made his choice), and change all delay times in these areas. Then, i realized that it is better to make a new variable, SPD, which is about the ball speed and will be defined by the player! We replaced all <for delay=1 to 31> with <for delay=1 to SPD>, so the players now can define the game speed as they want, depending on their reflexes, and enjoy most the game in the speed they want and it is better for them! (Players enter SPA, and then we define that SPD=60-SPA, and we suggest speed 25-30 which is delay 30-35).

Another problem we had has to do with goalkeeper disappear and re-appear in the new position. Amstrad is slow in drawing and due to this it was appearing as there was a pause during ball movement in that step. What we did is that we removed the delay (for delay=1 to SPD) for that step only. This helped a lot, although that there is still a small delay in drawing the graphics, but it is much better now, without affecting in a high degree the quality if the game.

Computer decision for shooting or keeper's move depend on random, using RND. At first, i used only RND but i realized that the decisions were the same if you restarted the emulator. This happened because we have to add the command RANDOMIZE TIME, so that the numbers don't have the same sequence. Also, it is necessary to define another variable when using RND in if /then commands. That happens because when you use RND in 2 lines, the number has changed! So, we define TYX=RND and we proceed the if/then commands using TYX, which is defined by RND. TYX then defines K (if <0.333333333, >0.666666666, or in the middle) and TYXG defines G. These subroutines are activated only if COM (in one player game) =2 (defined in the start) and we have to match it with the correct PLIK, and then using GOTO bypass inkey$ searching for player 2 keys.

Also, faces of kicker and keeper change to happy or sad depending on goal or save! This was easy, as we used chr$.

Two proposals for better game were the following: the first is that it would be better if kicker and keeper had more options (mostly high or low- eg right and up). But this we found very difficult to make. First, it is very difficult to show depth and make shadow of the ball when we use characters (chr$ and locate) for moving, as our choices of moving are very limited. If we used pixels only, maybe this could be possible. We could have used only pixels, but this would be very time consuming. In addition, I am very bad in painting and it is very difficult for me to understand how depth is designed in a picture or movement. Also, i don't know how to search inkey$ for 2 keys pressed simultaneously, as (when we search for them independently also), one of the two keys would be pressed first even with very small difference. One solution could be to define a new key (eg new key for up+left), but this i think would reduce game playability. The second proposal is to show the kicker make 2 steps before shooting. This is not bad option, but i thought that it is better to leave him not moving.

As for the keys, i know that player 2 keys are not the best option, but i wanted to be sure that everything is working ok in the real CPC, as i was afraid that there may be incompatibilities because i work on a emulator. So, i preferred standard letters to be sure that all will work ok.

In the list of the game, sections are usually named using REM command (there are both Greek and English), so that you can understand generally how the game goes on (for example it is mentioned K=0 G=1 shooting centre keeper right – what follows is activated only in the case that players have made this choice).

The game was developed using Winape Emulator, but also is ok for Java emulator- in Java we noticed a small but not bad difference in the brightness of the white colour, especially for the nets.

I know that my game is not of the best, as i know very little things about game programming, and i don't know how to use cpctelera or other engines that help. But i think that with more work and much thinking, i managed to make a game that is simple of course, but playable and can be enjoyable. For me, creating this game was a fantastic experience!

So, that's all the history of this game!

Hope you enjoy it!

Welcoming your comments at skafesakis@yahoo.gr

Many thx for reading this!

Sakis Kaffesakis

Heraklion Crete, Greece