

Basket Coach Board

Ladislav Ratgeber*

*Ratgeber Academy, Pecz, Hungary, ratgeber.laszlo@gmail.com

Abstract - Information technology development and its integration in all aspects of our social and economic life did not leave out the sports. Increasing professionalism and competition cause clubs to approach all their activities in more and more systematic way. Progress is especially visible in training and analyzing opponents' teams and players, i.e. scouting. Therefore a need arises for such a program package that would enable better knowledge regarding basketball tactics, action analyses and creation, review of video actions and printing reports, i.e. documenting all actions, results and remarks during the work. This program is used by national team of Serbia when preparing the games in all official and friendly matches. This paper describes design and realization of program package for analyzes, scouting and creating basketball actions. Graphic objects comprising the editor with their own drawing methods. To realize such editor it was necessary to realize a number of auxiliary methods as bitmaps operation (rotation, transparency, resizing, shrinking and widening by certain factors), methods for manipulation and drawing the actions (paths) assigned to players (drawing line with particular dots, calculating distances in pixels between dots, dividing lines in certain dots) as well as methods connected to analytic geometry (distance between the dot and the line, line crossing etc.). Basketball actions consist of phases with their own methods of drawing and animation. Animation is an iterative procedure where in every step, under certain conditions (speed, delay) players are moving from spot to spot obtained by "chopping up" the trajectory of the player.

I. INTRODUCTION

In team sports there is a number of methods to prepare athletes for competitions. There are physical, technical, tactical, psychological and integral ways of preparations. Every one of those has its fundamental importance in forming the athlete and the team, bringing to successful performance at the game and to the good result. A good and high quality scouting is unthinkable today without modern information technologies. This especially goes for the American football scouting. Every team has its so-called "whisperer", a person who reads lips of an opponent coach or player in order to anticipate next action. Nevertheless, human factor is still most important in a scouting. It doesn't matter whether scouts have read opponents action properly; the player in the field is the one to decide on last point, last good defense, and last foul... Scouting in volleyball is also interesting, especially in a way of its implementation. According to the best Serbian coach of all times, Serbian national team became European and World champion only when it modernized its information technologies. In basketball, a special place belongs to the "Advance scouting", one of best scouting programs in this field. It was first used by Chicago Bulls and they were six-time NBA champions. They has great

individuals in their team, including probably the best NBA player of all times Michael Jordan, they did not win a single trophy until they introduced the "Advance scouting" program. This paper describes the scouting program.. The project is modeled using UML specification. In static aspect of the system, the Use Case model and Class Diagram model were used, and for dynamical aspect the Activity Diagram was used. Activity diagram describes a process with its starting point (initial point) for action or activity necessary to describe a job or a function, and its final or ending point. By activity diagram we describe a process, step by step, with all conditions necessary for the next step, all the way to the ending point. The Use Case diagram enables defining primary elements of a system (participants, user) and the functions he performs. In this way it is possible to format a global picture of a system, which will be a foundation for all future steps in realization. The Class Diagram is the graphical description of a system comprising of classes, its attributes, connections and relations between classes. Class model helps more efficient implementation by clear definition of system aspects. Since present operative systems are highly interactive with the user, a natural choice is the graphic action editor where working space window will be the coach drawing board. Main advance of such application over the drawing board is the possibility to save action, print report with notes during the presentation in front of players, as well as saving actions in a video format with possibility to play those with various speeds. There are several types of editor in market fulfilling these demands. Here is presented a framework that may become universal for all graphic editors of the type, for all team sports (soccer, basketball, handball, etc.). This paper presents basketball action editor, but with some minor changes this may become the universal graphic action editor. Implementation is realized in programming package *Borland Builder 6* in programming language C++. For creating a video, open source API – *OpenCV* was used.

II. DESCRIPTION OF BASKET COACH BOARD

One of first steps in designing and modeling of a presented problem is the analysis [1,2] and specification of demands [3]. During the analyze process, it is important to obtain answers to the following questions:

1. The basic purpose of program package that is being implemented, i.e. essential demands of the user regarding functioning of the system. The application is a graphic editor, enabling the user to add elements on the working space window, to adjust properties, to add actions to every element. The diagrams also must be shown, as well as

their review by phases, and creating a video record and printing the report.

2. Hardware demands of the program package

Some teams worldwide use PDA devices. They are relatively small and easy to transport. Also they have relative good memory and possibility of writing on top of them.

On the basis of previous steps, it may be concluded that there is a pattern for formation of graphic editor model, as well as standards it must posses:

1. Application is **MDI** [4], i.e. enables working with several documents (and diagrams)

2. Every document consists of two toolbars (one for adding graphic elements on worktop and one with icons for manipulating the phases, animation and operations undo, redo and delete) and the worktop.

3 Before creating a new diagram, enable user to choose in which field the basketball action will be created (half field or whole field), which is the field orientation (horizontal or vertical), how much host and guest players the action will start, as well as graphic mode for depicting layers at the worktop (**type**: circle, triangle, shirt, and **color**)

4. Every graphic element has certain properties

a) Player: name, position, shirt number, size, angle regarding field orientation, status (host, guest), and mode of representation on the worktop. For every graphic element it is possible to make corresponding notes.

b) Ball: size,

c) Action (path): comment, line width, speed and delay of elements moving on the path,

d) Marker to mark part of the worktop.

5. Enable operations *cut*, *copy*, *paste*, *delete*, *undo* and *redo* on graphic elements:

a) Enable user to adjust paths where animations and pictures will be recorded, and to make adjustments for working with diagrams: animation sped, whether paths will be visible during playing the animation, player positions, as well as adjustments of values for element size on worktop – size of selection dots, b) Basketball action is happening in stages. After adding actions to players, moving to next stage is done by clicking the icon, and before next stage present one is being animated. Players in second and following stages are not able to move using mouse, in order to provide logic consistency of the action, c) Enable manipulation of stages, i.e. deleting and adding stages, moving to next one, previous, first and last stage, d) Enable animation of the action, i.e. showing animation of active diagram

6) Basketball action may be saved in *AVI* or *SWF* file. Enable playing these formats within the application.

7) Enable *screen capture* of active diagram

Generating two types of report: **general** – with user's comments, and **report of basketball action** with

characteristic pictures of action, i.e. all stages and all comments added by user.

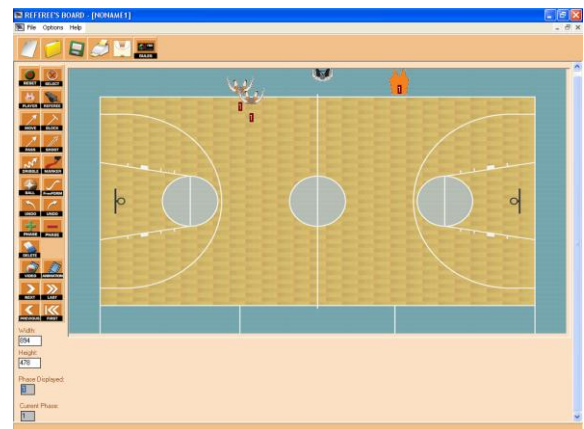


Figure 1. Application in the moment of action creating

If there will be no overlapping, that is the distance between players in next stage is less than set value, a message is written on the screen that user must re-check all connections assigned to players, so there would be no overlapping. If everything is fine, will be no overlapping on the screen. Afterwards the *DeselectAll()* method is being called, to deselect all elements. Figure 2 shows dialog for setting properties of the path. User clicks right mouse button on the path and chooses item Line Properties. Text entered by user is short commentary about player's movements, for printing in reports. Width parameter is thickness of the line by which a player is moving, and parameters Speed and Delay determine speed and delay of the player. In case of a referee and his assigned path, user chooses in which direction the referee will be moving and by which angle his view point is moving (in fact, referee's moving is determined by the path, and view point is determined by head movements according to the Target Point parameter).

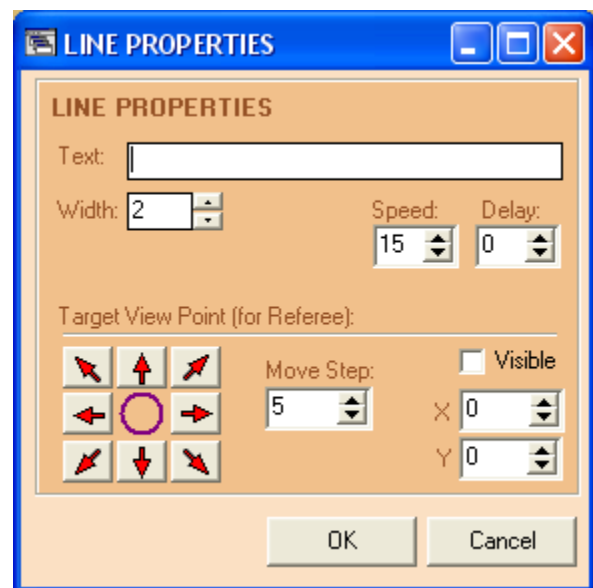


Figure 2. Dialog for setting properties of the path

The *CalculateNewPosition* method is being called, returning the dot where the element will be moved. In the

end, calling the *MoveTo* method provides movement of the object itself. When the element reaches the end, i.e. when *distance* is equal to value of *previousDistance* attribute, then *finishedAnimation* attribute obtains value *true* and animation is being finished. When playing the whole animation, a list of all stages is being checked which is the attribute of *TAnimatedAction* class and for every stage an *AnimatePhase()* method is being called.

Fig. 3 and Fig. 4 are showing the process of creating a basketball action.

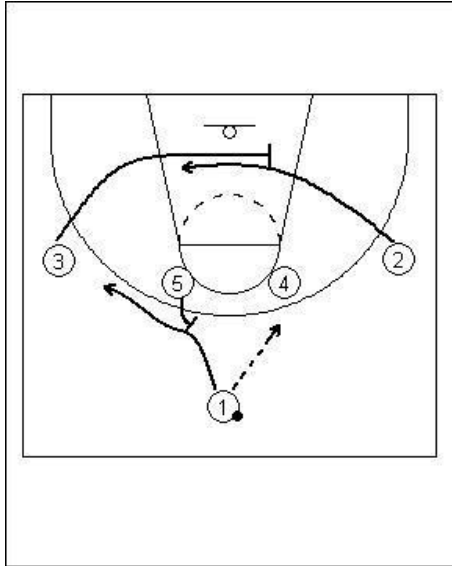


Figure 3. Example of basketball action

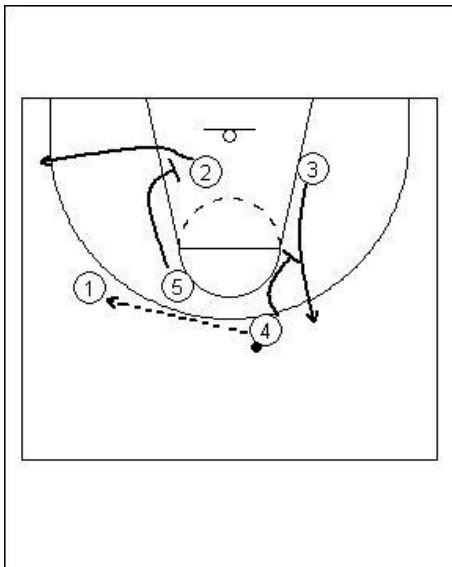


Figure 4. Example of basketball action

III. PRINTING A REPORT

For printing reports, components of *QuickReport* are being used. The basic component is *QuickRep* or the report that is to be printed. For every information or a group of data, there is a component *SubDetail* where data are placed. In short, every *SubDetail* must have its parent component, i.e. *QuickRep* component, and every

information must have *SubDetail* for a parent. Components are created dynamically since report depends on number of stages and the choice of the field. If user chose half field for creating a basketball action, in report four pictures are created in a same row, for every stage separately (picture represents starting state of every stage). Figures are obtained by calling *createBitmapsForPrint()* method. This method places a whole worktop (for every stage separately) into a list of pictures, which later will be dynamically created at certain positions. Every stage has *Bitmap* attribute, which is initial state of a stage. After that, listing all comments given by user for graphic comments is in order (if there are those). By clicking the right mouse button and choosing item properties, the following dialog is being opened.

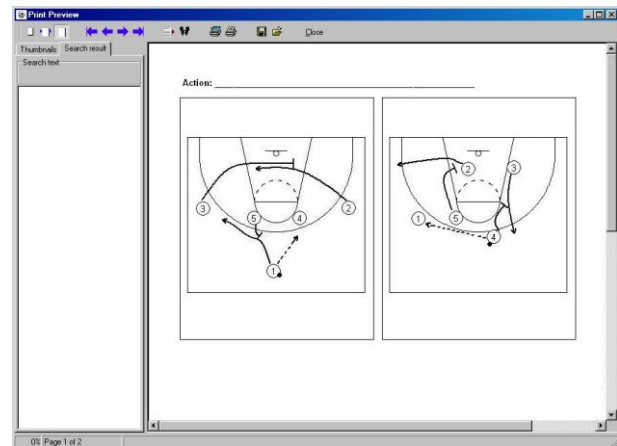


Figure 5. The example of the report

IV. CONCLUSION

This paper presents the implementation process for graphic editor for creating, editing, animation, and creating video for basketball actions. Steps from modeling to implementation are described. In every step, starting from model of Use Case diagram, class diagram, to dynamic aspect of a system, procedures are described that enable defining implementation issues of such an editor in a gradual and complete way. As it may be seen, this task demanded from our team to implement knowledge not only in modeling and coding, but also of the posts, its tactic element, parameters important to coach in scouting and tactical training of the player, and in adjusting user interface to final user. Basketball actions comprise of stages with their own methods of drawing and animation. In solving animation problems and animate stages it was necessary to learn how to handle lists of graphic objects, lists of distinct dots, as well as moving objects regarding user's adjustments: speed and delay. Largest problem was the way in which different speeds of players will be realized. Methods had to be written establishing current position of a player, his dots and on the given speed move him to strictly designed place. Before animation it was necessary to "divide" lines between distinct dots into more dots by which player may move. This was necessary in order to provide smooth animation, and speed problem was solved by moving a player for several dots in one iteration. This means that animation is iterative process, where in every step under certain conditions (speed,

delay) players move from one spot to another, obtained by „chopping up“ the path of the player. Possible improvements in this segment may be better algorithms for checking the lists or maybe a new data structure, which will speed up object manipulation. Aim of projecting and realization of such a problem is to enable basketball coaches that, together with players, in an interactive way, develop tactical skill for challenges waiting for them. On the basis of previously said, we may conclude that implementation of this kind of editor may be broadened in order to fulfill demands of other team sports, such as soccer, water polo or handball. Use Case model, class diagram and state diagram, with some changes, may become a pattern for creating specialized editors for these sports.

REFERENCES

- [1] Wiegers, Karl E. (2003). *Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle*, 2nd ed., Redmond: Microsoft Press. ISBN 0-7356-1879-8.
- [2] "Chapter 2: Software Requirements", in *Executive editors: Alain Abran, James W. Moore; editors Pierre Bourque, Robert Dupuis: Guide to the software engineering body of knowledge, 2004 Version*, Los Alamitos, CA: IEEE Computer Society Press. ISBN 0-7695-2330-7. Retrieved on 2007-02-08. "It is widely acknowledged within the software industry that software engineering projects are critically vulnerable when these activities are performed poorly." . March 2005
- [3] <http://www.techwhirl.com/techwhirl/magazine/writing/softwarerequirementspecs.html#comment-170>.
- [4] http://en.wikipedia.org/wiki/Multiple_document_interface.
- [5] A. Cockburn (2001). *Writing Effective Use Cases*. Addison-Wesley Longman Publishing Co., Inc. ISBN 0-201-70225-8.
- [6] Aurum A. Cox, K. and Jeffery. An experiment in inspecting the quality of usecase descriptions. *Journal of Research and Practice in Information Technology*, 36(4):211–229, 2004.
- [7] E. B. Fernandez and J. C. Hawkins. Determining role rights from use cases. In *RBAC '97: Proceedings of the second ACM workshop on Role-based access control*, pages 121–125, New York, NY, USA, 1997. ACM Press
- [8] R. Hurlbut. A survey of approaches for describing and formalizing use-cases. Technical Report 97– 03, Department of Computer Science, Illinois Institute of Technology, USA., 1997.
- [9] *The Unified Modeling Language User Guide* (Addison-Wesley Object Technology Series) Grady Booch, James Rumbaugh, Ivar Jacobson , 1998. Chapter II, III
- [10] *Introduction to UML 2 State Machine Diagrams* by Scott W. Ambler
- [11] Ball, R., Beck, J., DeMott R., Deneroff, H., Gerstein, D., Gladstone, F., Knott, T., Leal, A., Maestri, G., Mallory, M., Mayerson, M., McCracken, H., McGuire, D., Nagel, J., Pattern, F., Pointer, R., Webb, P., Robinson, C., Ryan, W., Scott, K., Snyder, A. & Webb, G. (2004) *Animation Art: From Pencil to Pixel, the History of Cartoon, Anime & CGI*. Fulhamm London.: Flame Tree Publishing. ISBN 1-84451-140-5
- [12] Solomon, Charles (1989). *Enchanted Drawings: The History of Animation*. New York.: Random House, Inc. ISBN 0-394-54684-9
- [13] <http://opencvlibrary.sourceforge.net/>
- [14] http://en.wikipedia.org/wiki/Graphical_user_interface
- [15] *Sams Teach Yourself C++ in 21 Days* (5th Edition) (Sams Teach Yourself) by Jesse Liberty and Bradley L. Jones (2004) , Chapter 14
- [16] *Borland C++ Builder 6 Developer's Guide* by Jarrod Hollingworth, Bob Swart, Mark Cashman, Paul Gustavson
- [17] Nicolai M. Josuttis. *The C++ Standard Library: A Tutorial and Reference*. Addison-Wesley. ISBN 0-201-37926-0.