Wanda: a framework to develop card based games to help motivate programming students

Rafael R. Drumond André L. Brandão Carlos S. Salles

Federal University of Maranhão (UFMA), Dept. of Informatics (DEINF), Brazil

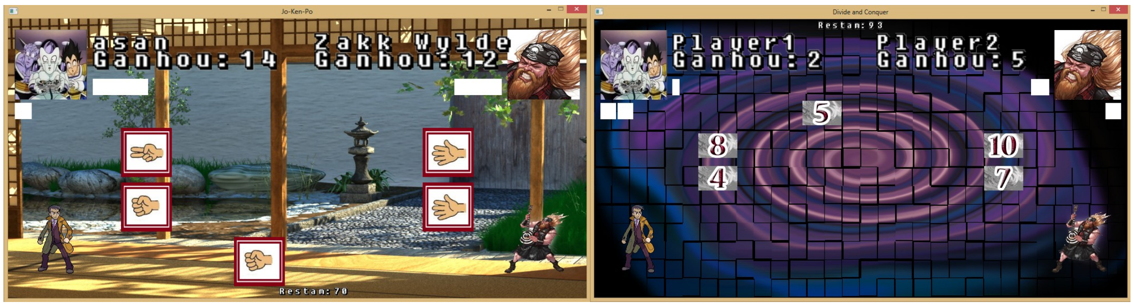


Figure 1: Educational card based games, Jo-Ken-Po to the left and Divide-and-Conquer to the right

Abstract

Teaching computer science, algorithms and other programming subjects can be stressing and difficult. Students motivation might drop due to the difficulty of learning how to program. To deal with this matter, this paper presents Wanda, a simple framework created to build specific card-based games that can be used as educational tools. With Wanda, professors can create educational electronic card games that require students to create virtual agents to play them. These games can help teachers and tutors to motivate programming students.

Keywords: card game, game framework, serious games, games for education

Authors' contact:
`{radrumond,andrelbd1,csalles}@gmail.com`

1. Introduction

One of the main issues around computer science undergrad courses in Brazil is that they are affected by a high drop out rate [Takahashi 2009]. This drop out rate is often related to the difficulty of learning programming skills [Kinnunen and Malmi 2006]. This turns out to be a disturbing issue since the number of open positions in the IT industry been increasing year after year [Celso Jr. 2009].

Programming teachers and instructors often find themselves struggling to find ways to attract students and motivate them while teaching and exercising algorithms knowledge. Teachers need to use intrinsic strategies inside the classroom to motivate students [Huitt 2011]. These strategies are related to the use of games, simulations with activities that stimulate logical thinking and allow students to stipulate their own objectives. An interesting approach can be found in [Bayliss and Strout 2006], in which it is encouraged the adoption of activities that students can relate to.

The objective of this paper is to present Wanda, a framework that allows computer science teachers and instructors to create simple digital card-based games (such as the ones illustrated in Figure 1) that require students to program agents to play the game. With these games, Wanda helps teaching basic concepts of programming while using intrinsic strategies to motivate students and helping overcome the difficulties found while learning how to program.

This paper is organized as following: Section 2 describes related work. Section 3 describes the concept behind Wanda and explains how it is structured. Section 4 gives examples of games that were created with Wanda and used inside of a freshmen classroom of a undergraduate computer science course. Section 5 concludes the paper.

2. Related Work

There is a vast literature on the development of games for educational purposes. Regarding this work, the most relevant related approaches are: (1) Using the Chien2D library to teach programming languages [Radtke and Binder 2010]; (2) Teaching programming languages using the PyGame library [Rebouças 2010]; (3) the S.P.A.R.S.E environment [Souza 2010]; (4) the learning impulsion through educational games [Kessler 2010]; (5) SDM, an educational game for software engineering [Kohwalter et. al 2011]; and finally, (6) an Educational Robotic Game for Transit Education Based on the Lego MindStorms NXT Platform [Silva and Magalhães Netto 2010].

The work done by [Radtke and Binder 2010] consists of using the Chien 2D game engine, a game engine with simplified functions and translated to portuguese, with a specific methodology designed to teach programming language to computer science students. By Challenging students to create games using Chien 2D it was possible to teach programming language concepts while allowing students to learn basics of game programming and to be motivated.

Using the PYgame library, [Rebouças 2010] made possible to establish a project to teach high school students how to create and develop games while increasing the interest of students towards computer science area.

The S.P.A.R.S.E (Software Project semi-Automated Reasoning tool for Software Engineering) project [Souza 2010] was developed to simulate a software engineering environment digitally. It is a game that has been used with a group of students who showed improved knowledge about software engineering.

The knowledge impulsion project [Kessler 2010] consists of creating educational games designed to improve and enhance practical and theoretical knowledge about portuguese, mathematics, chemistry and physics. These games are developed following two basic concepts: reviewing basic knowledge and building new ones. It was concluded that if well projected, the games generated could improve academic development.

SDM [Kohwalter et. al 2011] is a game that focus on teaching software engineering concepts similarly to S.P.A.R.S.E. With SDM, students can learn concepts of people management, involving training, assigning tasks, and workload in a practical way. The project managed to help a class of students to understand theoretical concepts taught in classes.

Finally, the Transit Education project [Silva and Magalhães Netto 2010] is a simulation game that uses a Lego built remote controlled robot in order to simulate urban traffic. The research team found that the game was both enjoyable and educational.

This work appropriates of concepts and ideas from the related works listed above. Instructors can use games created with Wanda to teach and motivate students easily by using an environment that students are to relate themselves with. Different from other works Wanda allows teachers and instructors to customize card-based games and use them as educational tools spending a small amount of time and effort to do it, while giving students a challenging and competitive environment.

Wanda has some advantages over the related works listed in this section. The work related to the use of Chien2D library, requires students to already know basic concepts about algorithms to be used, it is also time consuming and requires planning. Using Pygame to teach programming to highschool students consumes a lot of time and can't be executed as a short time homework to computer science students. S.P.A.R.S.E game is an interesting approach, however it is dedicated to teach software engineering and not algorithms. Knowledge Impulsion project aims to help teaching subjects such as portuguese, mathematics, chemistry and physics, but not algorithms. SDM and Transit Education projects also uses games in order to teach people management and driving skills respectively, but have scopes different form computer science. Under the scope of teaching basic concepts of algorithms, Wanda can help teachers with the task of teaching programming skills to beginners without consuming too much time and energy from both instructors and students.

3. Wanda Framework

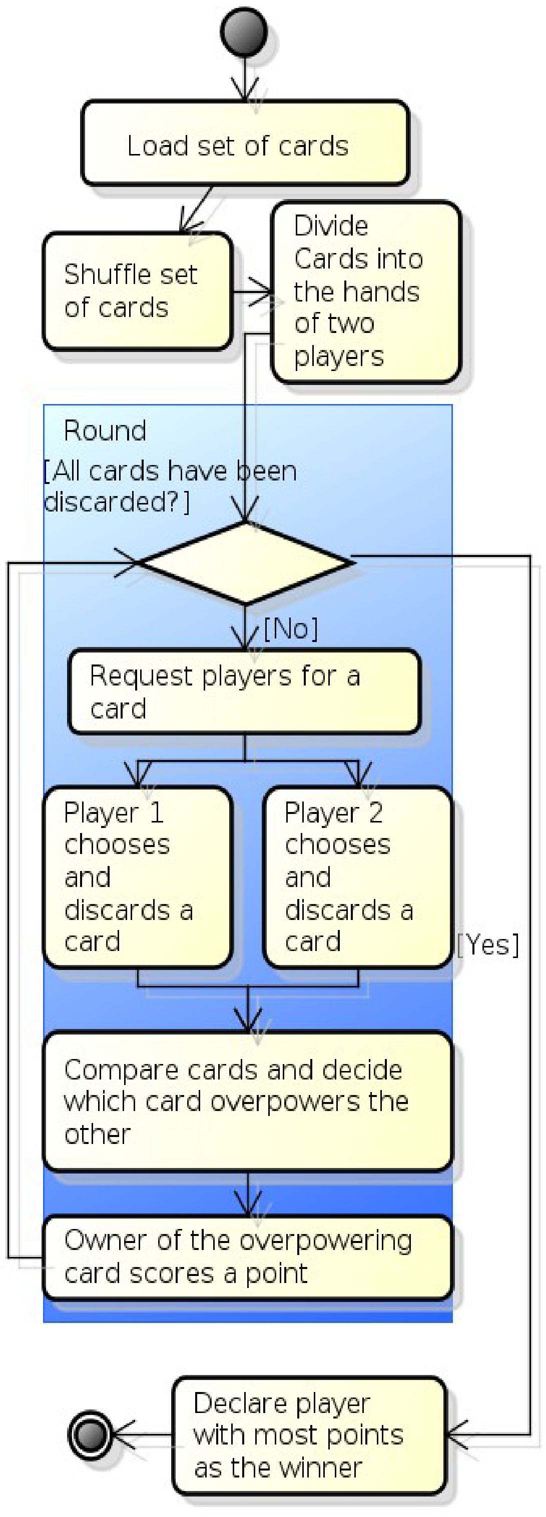
Wanda Framework offers teachers and instructors a high level of code reuse in the specification of new card-based games. Students have to develop new agents in these games. While implementing these agents, students exercise programming logics and algorithms.

Wanda Framework helps to develop games that follow the rules specified in Figure 2 by default. These rules are: At the start of every match, a specific set of cards is shuffled. This set is divided into two groups that will compose each player's hand. The game is played in rounds. At the beginning of each round both players choose and discard a card from their hand at the same time. The cards are compared according to the rules to decide which card overpowers the other one. The player that discarded the winning card scores a point. The game ends when all cards have been discarded, and the player with most points wins the match.

The job of the teacher is to provide the framework with the game rules by defining and creating: (i) the cards that will be used in the game; (ii) instructions to decide which card beats the other; and (iii) what kind of information the player will have access to. For example: the cards that are in his hand, or which cards have been played. Then, the instructor should hand the application to the students.

Finally, the students (players) should finish the implementation by writing the “strategy” inside the game code. A strategy is an algorithm function that receives the information provided by the instructors implementation and returns an action to be taken by the correspondent player (for example: what card will be used at that round).

Since the game is played by two players, two students need to implement a strategy each to compete with each other.


Load set of cards
Shuffle set of cards
Divide Cards into the hands of two players
Round
[All cards have been discarded?]
[No]
Request players for a card
Player 1 chooses and discards a card
Player 2 chooses and discards a card
[Yes]
Compare cards and decide which card overpowers the other
Owner of the overpowering card scores a point
Declare player with most points as the winner

.

Figure 2: Game flow diagram

3.1 Conceptual design

Wanda can be divided into three different parts: Mechanics and Interface; Rules; and Strategies. Each one has a different role in the final application.

"Mechanics and Interface" (MaI) is the core part of Wanda. The instructor and students do not need to acknowledge or change this part in order to use the framework. MaI controls the basic functions of the game, such as: removing and shuffling and distributing cards; displaying animations, scores and player visual information; controlling scores; and calling the rules specifications from the "Rules" part.

"Rules" group up the functions that define the rules of the game. The instructor is the one responsible for implementing this part. This part is called by the MaI in order to define the set of cards that will be used in the game. For every match that MaI starts, it will request an action from Rules at every round. At each request Rules receive the game state information and selects some of the information and sends to the Strategies of each player, which will return an action taken by each player.Rules must interpret the actions returned from the two different strategies and decide which action wins the round. The result should be returned to MaI. Then MaI will update the game state and animate the actions taken.

"Strategies" is the main focus of the application. The instructor should call a function from Strategies in Rules while sending it all the information visible to the player. The students job is to implement a function left in blank by the instructor (who will prepare all the parameters that will receive the information previously). This function must analyze the information received from Rules and decide which action the player is gonna take. The teacher should focus on how the information will be sent, for example: variables; vectors; arrays; tables; lists and others. As the purpose of this work states, the resulting game is destined to teach and motivate programming students, therefore the game should require certain level of programming skills based on the class average knowledge.

In Figure 7 the concept of MaI, Rules and Strategies, are represented by Wanda, Game Rules and Player Strategy respectively.

3.2 Implementation

Wanda was implemented in Lua [Ierusalimschy et. al 2014] using the engine Love2D [Löve Wiki 2014]. The engine was chosen since it uses Lua as a language which is simple and easy to use comparing to other languages such as python, java, c, c++ or c#, languages used by famous engines such as PyGame [Shiners 2014] and Unity [Unity Technologies 2014]. The possibility of leaving the code open to students, was another factor that favored the use of the Love2D engine. While leaving the code open, students can implement their strategy easily using Lua syntax as well as leaving the possibility of allowing his curiosity towards the game implementation open.

There are two Love functions that are used with Wanda. Draw and Update. Draw is a function where the game developer must call all functions that displays the interface inside the game. While Update, as the name says, keeps updating the game in real time.

The implementation was done based on the MVC (Model, View, Control) model. In order to divide Wanda properly, the functions of the View object were the ones that would be called inside Draw. As for control, would be the ones called inside of Update. The picture below (Figure 3) displays how this part of MVC was organized, following the UML notation.

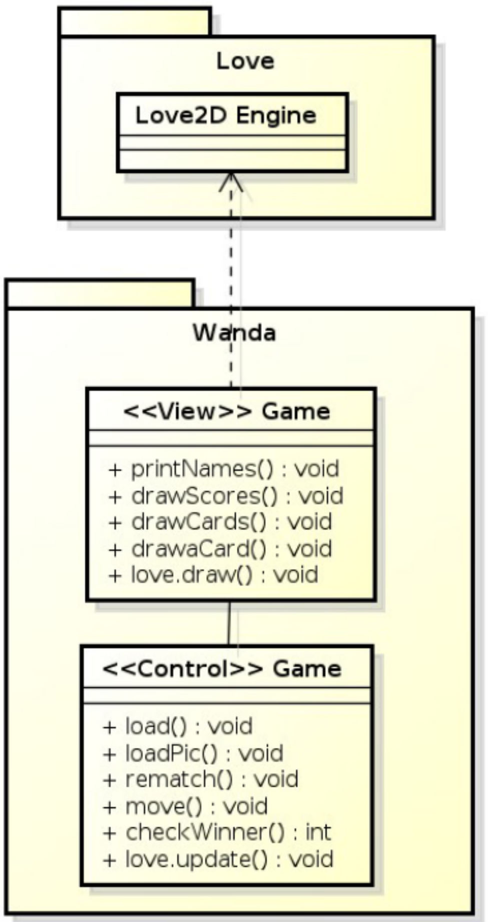


Figure 3: View and Control Diagram

As for the model part we defined a set of objects that would constitute the game. The game objects are described in figure 8, objects in blue represent the parts of Rules and Strategies, as described in the previous sub section. Each object represents an element of the whole game:

- Deck: Represents the initial set of cards that will be redistributed among players in the beginning of every match. It should be initialized in the rules function by making an array of cards.
- Card: Represents a single card of the game. It can be represented by anything.
- CardImage: Represents an image that will be displayed for each card depending on the card value.
- Player: Represents one of the two players of the game. It contains the images that will represent it in the game. It also holds the function of the students strategy that must be implemented by him/her.
- Hand: This object groups the cards that each player holds.
- HandCopy: this object is a copy of a players hand. Its purpose is to be used as a “dummy” in order to help guarantee stability in the access of the player hand by the student algorithm.
- Rules: It is the object that should be implemented by the teacher or instructor. It holds functions that will set-up the game, control the flow of events of every round and how cards will interact against each other.
- Game: It's the object that holds information that is common for every game. It contains information of the game such as scores, the stage of the game and animation variables.

Figure 4: Tournament of the Jo-Ken-Po game

4. Test Case

In order to test the framework, two games were created and given to a class of computer science freshmen students. The students created strategies and competed against each other in a small tournament (Figure 4).

The games created were very similar, but each required different programming skills and were applied in different moments. This section will describe both games.

4.1 Jo-Ken-Po

Jo-Ken-Po (Figure 5) is inspired on the usual Rock-Paper-Scissors. This game has six cards: Two scissors; two rocks; and two papers. They are shuffled and divided for two players. At each round both players must discard one card at the same time. The player that discarded the card that "beats" the other one is declared the winner of the round. Notice the comparing rule: rock beats scissors; paper beats rock; and scissors beats paper. After 3 rounds, the player with more round victories is declared winner.

In order to implement this game, it was necessary to start the deck with the six cards each one defined as a string holding the name of the card. At every turn each card of the player and the opponent hand is passed to each player as variables. Each player must return a string of the card he/she wants to use. After each strategy has been consulted, the cards are compared and the winner of the round is decided. After 3 rounds the match winner is decided. After a certain number of matches (specified by the instructor) are played, the player with most match victories is declared winner.

The purpose of Jo-Ken-Po is to teach students how to compare information on different variables and how to use "If" clauses. This can be achieved since the parameters are only variables and in order to the player find out what cards are in his/her hands he must compare the variables with the strings "rock", "paper" and "scissors" and deciding what card he must play based on the cards on his/her hands.

This game is supposed to be used with students with one month of studies in computer science since it only requires knowledge on if-else statements.

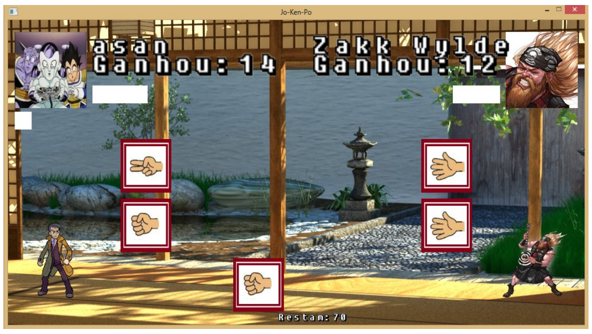


Figure 5: Jo-Ken-Po game

4.1 Divide and Conquer

Inspired on the card-based game Divide and Conquer (Figure 6). The game has ten cards: nine cards numbered from two to ten and a card numbered as 12. The game is very similar to Jo-Ken-Po, but with the difference that in order to win a round the player that plays the smaller card wins the round if he plays a divisor or predecessor of the bigger card, or else the owner of the bigger card wins the round.

This game was implemented in a similar way as Jo-Ken-Po, but instead of sending to the player strategy function his/her cards and his/her opponents as variables, it sends then as arrays along with the number of cards in each player hand. Players must return the index of the desired card in the array in order to discard it.

The Divide and Conquer objective is to teach students how arrays work, how to make iterations on them, how to extract information and what the array index means. This can be achieved since the cards on each player hand is passed to the player an array of numbers representing each card. The player must iterate and check the array values to find out what cards are on his/her hands, allowing the student to exercise knowledge about the subject.

This game should be used with students by the end of their first semester studying computer science. Students should have basic knowledge on arrays and iterations making possible to join this activity.

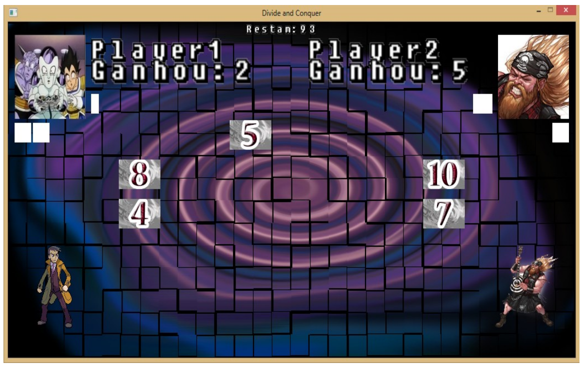


Figure 6: Divide and Conquer game

5. Results

With Wanda, it was possible to create the games Jo-Ken-Po and Divide and Conquer. These games were given to a class of students at different times. The students were requested to write strategies (for each game) to compete between them. After each game, students had to write a small paragraph about their opinions about the two games and answer a question stating how much they enjoyed the activity.

From the answers given, only 5% of the students did not enjoy the activity. 90% of students said the activity was interesting and fun. One interesting comment was about the game being a fun way of testing the knowledge acquired inside the classroom. The most interesting comment, pointed out that the activity helped the class to see that algorithms could solve more kinds of problems than they had imagined.

Divide-and-Conquer was considered more difficult than Jo-Ken-Po, but challenging at the same time. The questionnaire revealed that only 9% of the students were not so interested in the activity. 20% of the

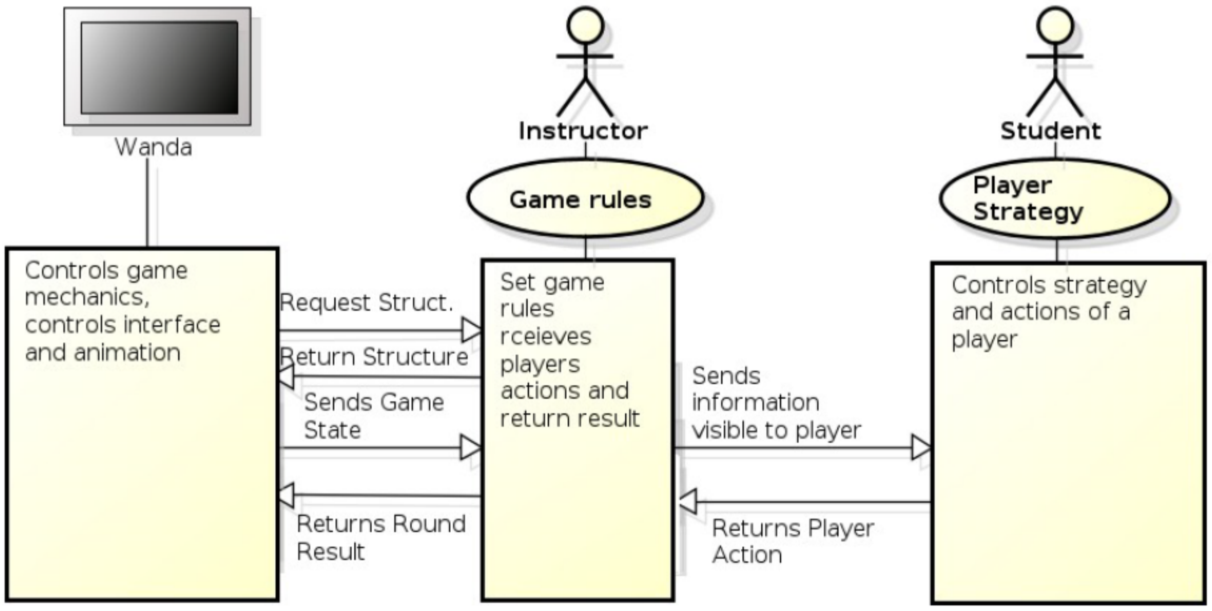

Wanda
Instructor
Game rules
Student
Player Strategy
Controls game mechanics, controls interface and animation
Request Struct.
Return Structure
Sends Game State
Returns Round Result
Set game rules rceieves players actions and return result
Sends information visible to player
Returns Player Action
Controls strategy and actions of a player

Figure 7: Conceptual Design diagram

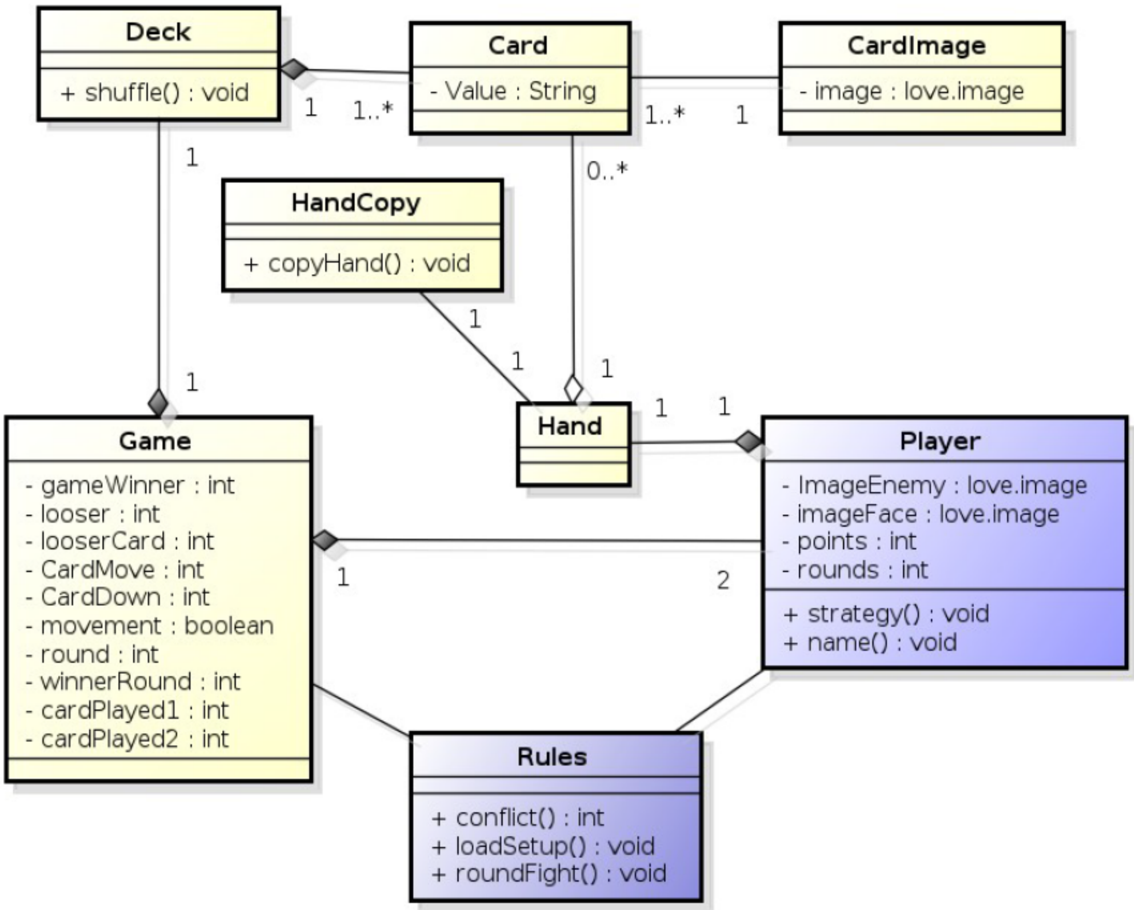


Figure 8: Object diagram

students stated that the activity was more difficult or challenging than the previous one. The most important comment about this activity pointed out that the game helped the students to visualize their algorithms operating and working.

6. Conclusion

With this work, it can be concluded that Wanda is a useful tool to create simple educational card games or hasten the creation process. Since the framework takes care of most of the mechanics, the teacher or instructor will only have to enter a small piece of code, images and format the player files. In order to generate a small game.

Although the example games presented on this paper look similar, Wanda still can create different kinds of card games. It's important to remember that the main point is not to create a different mechanic, but different rules and different ways to interact with the game in order to make students exercise different programming concepts inside class while creating a competitive and fun environment.

Another important point is the fact that the resulting game code is left open so that students might look into it and learn more about programming or even allow teacher to improve the tool to their needs.

As future projects, Wanda can be further used to develop more complex games, while creating a repository of games to help other instructors.

Acknowledgements

The authors would like to thank the students of Algorithms 1 that took part on the experiments of this project and were eager to collaborate. They also would like to thank Department of Technological Innovation for funding this research. The authors would like to thank CNPQ for also funding this research.

References

BAYLISS, J.D. AND STROUT, S., 2006. WOODCOCK, S., 2001. Games as a ”flavor” of cs1. *In: Proceedings of the 37th SIGCSE technical symposium on Computer science education.* 500–504.

CELKO, J. AND MCLEOD, J., 2005. Divide and Conquer. Available from: http://www.pagat.com/tile/wdom/divide.html [Accessed 20 June 2014].

CELSO JR., L., 2009. Setor de TI aponta falta de mão-de-obra qualificada. Available from: http://bit.ly/bQnsOg [Accessed 20 June 2014].

HUITT, W., 2011. Motivation to learn: An overview. *In: Educational Psychology Interactive, Valdosta, GA*. Available from: http://www.edpsycinteractive.org/topics/motivation/motivate.html [Accessed 20 June 2014].

IERUSALIMSCHY, R., WALDEMAR, C. AND FIGUEIREDO, L.H., 2014. Lua: the Programming Language. Available from: http://www.lua.org/ [Accessed 20 June 2014].

KESSLER, M.C., ET AL 2010. Impulsionando a aprendizagem na universidade por meio de jogos educativos digitais. *In: Anais do Simpósio Brasileiro de Informática na Educação*.

KINNUNEN, P. AND MALMI, L., 2006. Why students drop out cs1 course?. *In: Proceedings of the second international workshop on Computing education research.* New York, NY, USA, 97-108.

KOHWALTER, T.C., CLUA, E.W.G., AND MURTA, L.G.P., 2011. SDM – An Educational Game for Software Engineering. *In: Anais Simpósio Brasileiro de Jogos e Entretenimento Digital*.

LÖVE2D, 2014. Löve Wiki. Available from: http://love2d.org/wiki [Accessed 20 June 2014].

RADTKE, P.V.W, AND BINDER, F.V., 2010. Chien 2D: A Multiplatform Library to Teach the C Language Through Games Programming. *In: Anais Simpósio Brasileiro de Jogos e Entretenimento Digital*.

REBOUÇAS, A.D.D., ET AL 2010. Aprendendo a ensinar programação combinando jogos e Python. *In: Anais do Simpósio Brasileiro de Informática na Educação*.

SHINERS, P., 2014. PyGame. Available from: http://www.pygame.org/wiki/index [Accessed 20 June 2014].

SILVA, M.M., AND MAGALHÃES NETTO, J.F., 2010. An Educational Robotic Game for Transit Education Based on the Lego MindStorms NXT Platform. *In: Anais Simpósio Brasileiro de Jogos e Entretenimento Digital*.

SOUZA, M.M., ET AL 2010. SPARSE: Um ambiente de Ensino e Aprendizado de Engenharia de Software Baseado em Jogos e Simulação. *In: Anais do Simpósio Brasileiro de Informática na Educação*.

STROUSTRUP, B., 2010. Viewpoint what should we teach new software developers? why?. *In: Communications of the ACM, 53 January 2010 New York*. New York: ACM Press, 40–42.

TAKAHASHI, F., 2009. Matematica e ciências da computação têm alta taxa de abandono. Available from: http://bit.ly/daPZUm [Accessed 20 June 2014].

TUAN, H.L., AND SHIEH, S.H., 2005. Mthe development of a questionnaire to measure students motivations towards science learning. *In: International Journal of Science, 639-654*.

UNITY TECHOLOGIES, 2014. Unity Game Engine. Available from: http://unity3d.com [Accessed 20 June 2014].