THE PRIMEGAME: COMBINING SKILLS IN UNDERGRADUATE COMPUTER SCIENCE PROGRAMMES

Jens Fendler[1], **Manfred Meyer**[2]

[1] Polytechnic of Namibia
Windhoek / Namibia
jfendler@polytechnic.edu.na

[2] University of Applied Sciences Gelsenkirchen
Bocholt / Germany
manfred.meyer@fh-gelsenkirchen.de

Abstract

Foundational software engineering skills such as numerical methods, algorithm design and programming are usually taught as disjunct courses in most undergraduate computer science programmes. While those and other skills have to be combined in order to develop suitable solutions, such integration usually only takes place in advanced courses where larger projects are then often subject to peer-based work.

In this paper we are proposing the PrimeGame as a novel game-based programming tool which combines a wide variety of computer science skills in a small and manageable environment.

The PrimeGame is a simple two-player board game which, however, is not fully-enumerable and thus can not be solved with pure computational power. Students are required to implement autonomous players for the game, in an attempt to model their own ideas on how to play the game best and to eventually beat the opponent. All submitted players will finally compete against each other in a tournament setting. As both, a simple player interface and a comfortable and fully functioning testing environment are provided, students don't have to care about the game engine, GUI components, or any other advanced aspects. They can instead focus on the design of suitable algorithms, as well as syntactically correct implementations thereof. The PrimeGame testing environment as well as the interactive graphical tournament framework including various on-line board presentations and ranking schemes are currently implemented as a Java project, thus student's players can easily be provided as Java classes. Common wrappers and interfaces for players developed in other programming languages are available as well.

In developing PrimeGame players, students get early exposure to a combination of skills as necessary for all major computer science (CS) projects: from initial ideas and intuitive algorithms over studies of performance aspects, formal representations, implementation, up to evaluation and testing are all combined in a game project that can be individually completed within a few weeks time.

Using the game in class, we witnessed an increased student interest in the subject matter, as well as a lot of fun among participating students. As the PrimeGame approach has so far been used and evaluated in undergraduate CS courses in not only European but also in African environments, it exposed different cultural attitudes towards competitive learning.

While so far we have mainly focused on undergraduate aspects of the game, we are currently looking at advanced concepts which could be taught similarly well in graduate and post-graduate programmes. These include autonomous and co-operative agents, advanced data structures and algorithms, artificial intelligence and advanced numerical methods.

Keywords - Algorithms, competition, competitive learning, complimentary skills, computer science, cultural aspects, education, games, Java, numerical methods, prime game, programming.

1 BACKGROUND

In most undergraduate CS programmes, basic skills like mathematical modelling, algorithm design, programming and complexity considerations are usually taught as part of different courses [2, 5, 7]. For the development of suitable solutions, these and other skills need to be combined. However, such combination often only takes place in advanced courses, focusing on larger and more complex projects, which then have to be developed and evaluated as teamwork. Based on previous success in game-based learning as shown e.g. in [6], [8] or [9], this observation led to the design and use of an easy-to-understand but rich-enough two-player board game for which students are asked to develop computer implementations of their individual strategies which will eventually compete against each other in a final tournament.

The history of this so-called “PrimeGame” (a name that becomes obvious later when presenting the rules) dates back to the year 2003 when we first used this game-based learning approach in first-year CS courses at the University of Applied Sciences Gelsenkirchen. In fact, the PrimeGame was introduced in class after very few weeks of introductory programming lectures.

While students were almost instantly able to develop and apply basic game strategies when playing the game “manually" in class between two student groups, this turned out to be more challenging for them on the PC. Based on fundamental data and control structures, as were previously introduced in theory lectures, students now had to formatlize their respective strategies, abstract formulas and algorithms into executable program code. Nevertheless, the initial task of developing working players was simple enough to be managed by students in little time, thus providing almost instant positive feed-back and encouragement for further discussions and algorithm refinements.

By also providing a framework for testing player implementations and running competitions, students became highly motivated to not only deliver formally correct players, but also to develop and improve their strategies in order to come up with the best performing player for the final PrimeGame Competition which still usually takes place the week before Christmas.

In 2007, the PrimeGame approach was then used for the first time as part of the second-year course in Object-Oriented Technologies (OOT) at the Polytechnic of Namibia. Due to different course objectives, the focus shifted from basic strategy implementation using primitive control structures and data types (covering mainly formalization and implementation aspects of applicable strategies) towards more elaborated topics on Object-Oriented software design. Among others, these include interfaces, inheritance and class libraries, as well as complexity considerations in order to meet given time constraints per move.

Although experiences from Germany and Namibia regarding the learning outcomes were found to be quite different (see chapter 6), in both cases – thanks to the meanwhile very powerful and illustrative Competition GUI – the final PrimeGame Competition is regarded as one of the semester’s highlights and continuously demonstrates that programming and the design of efficient algorithms can also be a highly entertaining endeavour.

2 THE PRIMEGAME

The PrimeGame is a two-player board game, based on a very simple set of rules which can informally be stated as follows:

1. The *board* contains all natural numbers from 1 to some limit *n* (board size may vary but be constant for the tournament). Both players’ scores are initially set to 0.
2. The *players* make alternating moves until all numbers have been taken off the board.
3. A *move* is selection of one of the remaining numbers on the board. When making the move, the score of the player will be increased by the chosen number while the score of the opponent will be increased by the sum of all remaining factors of the chosen number still being available on the board. Finally, the selected number as well as all its factors are taken off the board and the other player is to make the next move.
4. The *winner* is the player with the highest score at the end. For fairness, every one game should consist of two *matches*, giving each player the first move in exactly one match.

Box 1: The PrimeGame Rules

These rules shall be illustrated using a small example: Assume a board size of 20, thus in the beginning the board contains all natural numbers from 1 to 20 (inclusive):

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Box 2: The Initial PrimeGame Board for board size n=20

The player making the first move is now to select any number from the board. Obviously, some choices are more advantageous than others, e.g.:

- Selecting number 19 will increase the player's score by 19 while only adding 1 to the opponent's score. Thus the player's effective gain (score difference) is 18.
- Selecting number 20 instead would increase the player's score by 20. On the other hand, as the set of factors of 20 on the board is {1, 2, 4, 5, 10}, the opponent's score will be increased by 22 (the sum of all factors on the board). Thus the player's gain for this move is negative: -2.

Obviously, number 19 seems to be a much better choice than 20. Of course, choosing any other number (from 1 to 18) would result in a legal move as well. Yet, as can easily be seen none of them would result in an equal or a better gain (the so-called *direct advantage* of the move).

After having selected number 19, the updated board as presented to the opponent to make his move will be the following:

	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		20

Box 3: The PrimeGame Board after execution of the first move selecting number 19

Obviously, this sample match will terminate quickly after a theoretical maximum of 10 moves for each player. However, with increasing board size not only does the number of moves for each game increase, but also the complexity of the task to select a "good" number (i.e. the player's *strategy*).

2.1 PrimeGame Strategies

A number of trivial strategies can easily be defined – and possibly be used as reference strategies to evaluate the performance of students' players. Following, we are briefly introducing a selection of the most basic strategies, which have been used for evaluation purposes in the past:

- The *"Anxious Strategy"* always selects the smallest available number on the board thus leaving no factor for the opponent.
- The *"Greedy Strategy"* always selects the highest available number on the board not caring about factors at all. Selecting number 20 in the introductory example is an instance of applying the Greedy Strategy.
- The *"Prime Number Strategy"* selects the highest available prime number on the board. If there is no prime number left, it turns into the Greedy Strategy.
- The *"No Factors Left Strategy"* selects the highest available number not having any factors left on the board. A variant of this strategy applies the Prime Number Strategy for the first move and thus avoids selecting number 1 in the first move.
- The *"Best Advantage Strategy"* selects that number leading to the best direct advantage, i.e. the highest gain (maximization of the score difference).

Although the latter performs a local optimization, all of the above strategies have been proven not to be optimal for sufficiently large board sizes. More successful strategies require a more sophisticated analysis of the consequences of each possible move for later selections – for the opponent and the player itself. While such game-tree analysis can be fully done for small board sizes or situations with only a small set of numbers left, its applicability becomes drastically limited as the number of possible moves increases.

Currently, a number of different player implementations exist which all show a better performance than the Best Advantage Strategy. These for example focus not only on the current move but also take possible counter moves into account, thereby optimizing the score increment after both: the own plus the opponent's move (expecting a best direct advantage for the combination of both).

However, a thorough number-theoretic analysis has not yet been completed and is assumed to be a non-trivial problem. It is therefore still an open question if a general-purpose optimal strategy exists for arbitrary board sizes and set-ups, how it would look like, and whether it has a manageable run-time complexity. While an “optimal strategy” can obviously be found with a brute force approach, such is hardly appropriate for reasonable board sizes as it would easily violate given time limitations per move.

2.2 Practical Implications

In order to avoid strategies being developed and optimized only for specific game instances (e.g. for a fixed board size), it becomes crucial to decide about the board size to be used in the tournament *after* all players have been submitted, e.g. by using a random-number generator. For fairness reasons the board size should evidently remain constant for all matches throughout a tournament. If the number of participants in the tournament is small enough, a variant would be possible where a small selection of some again randomly chosen board sizes, e.g. a small, medium and a larger board size, is used.

While of course the board size itself can be any natural number, sizes ranging between 50 and 500 have shown to yield best results for classroom and competition environments - partly depending on the computer hardware used. Such boards are generally a good compromise between (intended) game complexity (and thus challenge) for the number selection task, and a reasonable run-time behavior allowing an audience to keep track of a complete tournament of up to approximately 150 players in separate rounds.

As we currently employ a round-robin approach during competitions in order to ensure every player faces every possible opponent exactly twice, groups of approximately 20 players still need to be formed as to limit the steep combinatorial increase in the number of overall matches.

2.3 Similar Board Games

By design, the PrimeGame is a light-weight game. Compared to popular traditional board games like Chess or Go, the PrimeGame features a much smaller set of rules, which makes it very easy to explain (e.g. in written project descriptions) and for students to understand.

Although the board is sometimes thought of as 2-dimensional, such notion does not reflect the nature of the game. It is essentially a 1-dimensional game, but – unlike other board games – can be easily represented in a variety of formats. This flexibility is actually useful in terms of algorithm design, in that data structures can easily reflect students' custom perceptions or ideas.

Due to the flexible -- theoretically unlimited -- board size, the PrimeGame search tree can easily become much more complex than the one for Chess. It is in fact comparable to other strategy games using scalable board sizes like Go or Othello/Reversi. While average board sizes as mentioned above are probably most flexible and applicable throughout a large number of courses or projects, huge boards may be useful e.g. for PrimeGame projects in courses on distributed computing. Thus the flexibility in terms of board sizes without an influence on player behaviour or expected success rates makes it superior to many other games often used for specific programming projects.

3 PRIMEGAME FRAMEWORK AND DESIGN

Currently, the PrimeGame framework features a highly modular design. It is composed of three main components: First, the Player interface(s) - defining valid player objects throughout the framework - which students must implement their players against. Second, the core Engine which sets up the board for each game, invokes the player objects' methods to take turns, to enforce the current ruleset, and to notify other components of game progress. And third, the Graphical User Interface (GUI) component which allows an easy tournament operation as well as visualisation of ongoing games. Among others, the GUI component provides different board visualisations, statistics, current scores and ranking tables. It further allows the operator to reduce game speed by requesting delays in the engine during two moves. This allows the audience to closely follow a game as well as commentators to announce intermediate results or upcoming games to the audience.

In addition to these modules - all of which are necessary in the competition environment - a simple testing component has been developed which is distributed to students along with the Player interfaces. This testing component, called Trainer, is a simple replacement for competition GUI front-end and allows students to test their players during development. It invokes the engine, but does not

enforce as strict rules (e.g. timing constraints). It handles run-time exceptions more lenient in that it informs the user of such problems, and rather tries to recover from player errors whenever possible in order to let the player actually complete the game rather than immediately disqualifying the respective player.

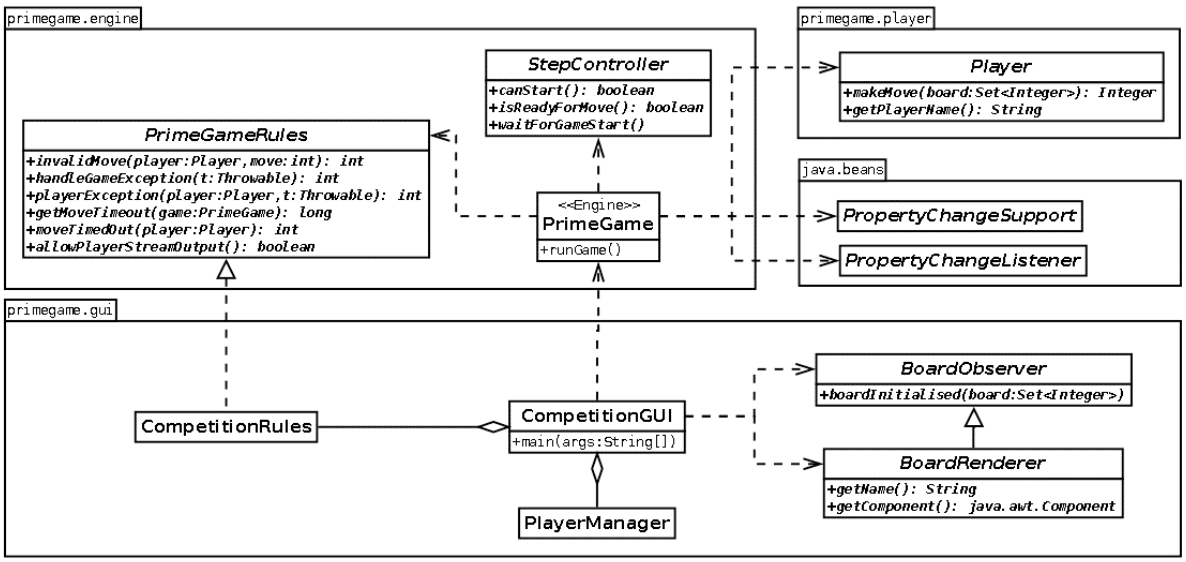

Fig. 1: The PrimeGame Framework Design

3.1 Player Interface

The core Player interface is kept very simple. It primarily consists of two methods to be implemented: the makeMove() method which receives the current board state as a set of integer numbers and must return a valid number representing the player's move. The second method, getPlayerName(), is used to query the visible name of the player in order to display it during games and in the ranking table.

Additional interfaces, inheriting from the Player interface, add methods to retreive the student number, as well as group identifiers to allow for automated group-based competitions. Currently, the GUI component also utilizes the student number (if provided through the corresponding interface) to load an image of the player provided in a file with the student number as a filename.

Depending on the intended environment and envisaged outcomes - e.g. an under-graduate course focusing on algorithm design - other interfaces may be used, requiring the students to implement either more or less complex players.

For students not yet familiar with the Java collection framework, a simple wrapper class around the Player interface has also been developed. This wrapper converts the passed set of integer objects in the makeMove() method into an array of primitive integer values, and requires a primitive integer value as its return value. This way students can implement their players with virtually no knowledge of OO principles or Java API specifics, and are enabled to more or less directly map pseudo-code algorithms into working player code.

3.2 Engine

The engine is the central component of the PrimeGame framework. It contains the complete game logic and is responsible for creating clean boards, enforcing the current rule set, invoking the players' makeMove() methods, and tries to detect any cheating activity by the players as good as possible. Using an event-based notification pattern, other components can register themselves with the engine as listeners in order to be notified of all relevant activities while a game is being played. This is e.g. true for the GUI and Trainer components, which continuously display the game state to the user. In addition, one component can register with the engine as a controller. This controller can then set competition parameters such as the board size, load different rule-sets, as well as start, stop and delay individual games. However it should be noted that the engine is not useable as a stand-alone application, but must be invoked from a separate front-end.

3.3 Competition GUI

The PrimeGame's Graphical User Interface has been designed as a traditional Java Swing desktop application. It integrates all framework components and allows the operator to control the various aspects of a tournament in which multiple players participate. The GUI is meant to be displayed on a large screen or projector, in order for an audience to follow competitions live. While the main functionality of the GUI is to provide an overview of the current game between two competing players, as well as to display the competition's ranking table including each player's score, it also registers a set of board renderers which give a comprehensive view on the current board, numbers taken, and available score. Additional renderers currently being developed will display statistical information, or provide a more "fancy" 3D view of the board.

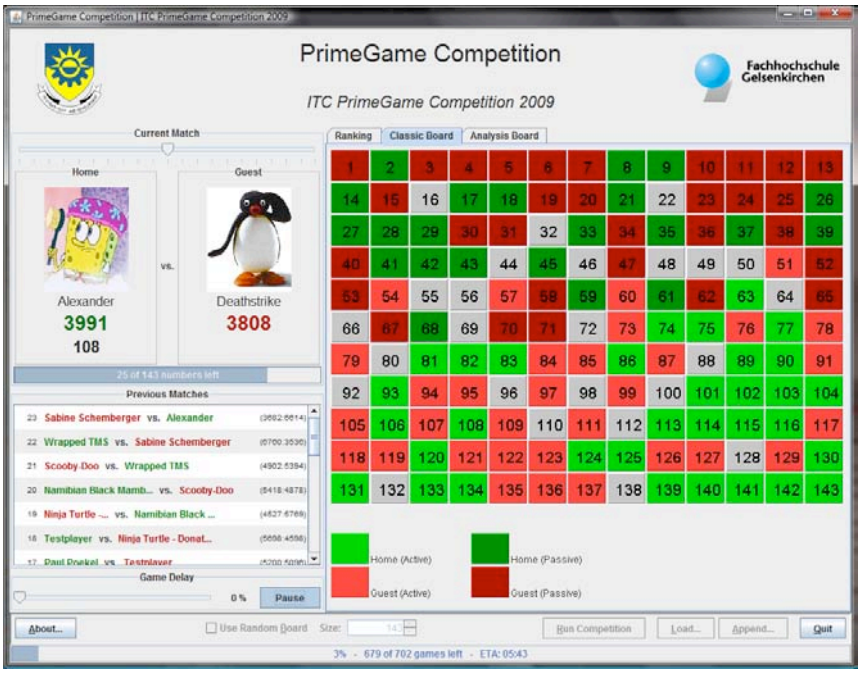

Fig. 2: The PrimeGame Competition GUI with “Classic Board” view

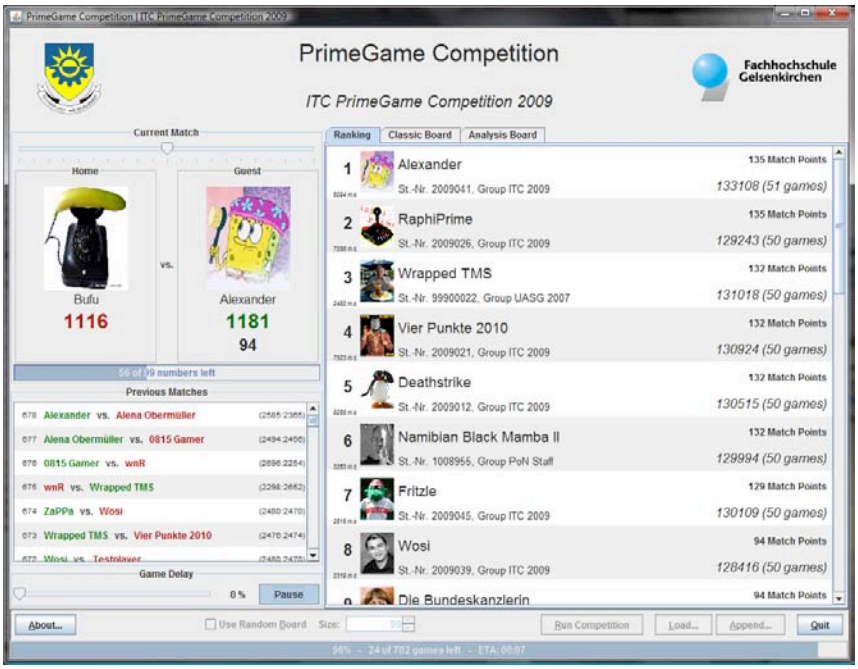

Fig. 3: The PrimeGame Competition GUI with “Ranking” view

3.4 Trainer

The Trainer is a very simple console application, which acts as a front-end and is distributed to the students for Player development. It neither imposes special resource constraints (such as a graphical console, or a certain screen resolution), nor does it enforce very strict rules during games. It allows students to let their Players compete against one other player while running the games as fast as possible. If desired, students can modify the Trainer code to register custom listeners for the Engine as to debug their players during a game. The console aspect of this simple front-end allows the Trainer to be scripted and thus run automatically (e.g. from within the student's build environment and tools) in an attempt to transperently blend into the Player development process with as little manual intervention as possible. While the Trainer can be setup to employ the competition rule-set, it uses a more lenient rule-set by default. Using this training rule-set no timing-constraints (i.e. maximum time allowed per move) are enforced, and exceptions thrown by a Player are tried to be handled as gracefully as possible in order to let the Player continue a game whenever possible.

4 PREREQUISITES / REQUIRED PRIOR LEARNING

In order for the PrimeGame to be successfully employed as a teaching tool, certain prerequisites must be met by the students. Clearly these requirements greatly differ depending on the intended learning outcomes, course level and the specific aspects of the game being studied. Yet we believe that the PrimeGame's potential is limited if used as one of the very first projects (i.e. early in the first semester). It was, however, successfully used as one of the final projects in first semester courses, effectively combining many of the skills students had acquired during their initial term.

As player development aspects will obviously be a primary aim in most CS teaching and learning scenarios, students should already have a fundamental understanding of certain key principles. First, a sound knowledge of program control structures, and foundational data types is obviously essential. Furthermore, basic abstraction skills are required in order to develop algorithms which will perform reasonably well on different board sizes, as well as against a variety of opponent players. As the PrimeGame - unlike traditional board games - leaves plenty of room for modifications (e.g. board size, number of participating players, etc.), students should try to develop as adaptive solutions as possible. Although the fundamental mathematics behind the game idea (or rules) is a rather trivial one, students will greatly benefit from a sound theoretical background in mathematics. Finally, apart from the ability to actually implement a given algorithm in e.g. the Java language, students should have a basic understanding of complexity theory and thus the run-time behavior of their solutions.

As far as group projects are concerned and students are asked to develop players in teams, it might be beneficial for the students to have some experience in project work and task distribution. However, in the context of the PrimeGame we believe those aspects can largely be neglected and won't impose any constraints due to reasonably small project sizes.

Other requirements in terms of prerequisite skills mostly depend on more specific aspects beyond the scope of player development. As the PrimeGame framework offers quite a variety of possibilities for extensions and customization, individual prerequisites will obviously differ accordingly.

5 ENVISAGED OUTCOMES

Over the past years, the PrimeGame has proven to be a powerful tool not only for supporting the learning process in introductory or basic programming courses. While originally being used in the very early stage of programming education, there is a huge variety of advanced aspects that can also be studied very effectively using the PrimeGame setting.

Depending on the course the PrimeGame is used in, the focus in an OO course may be more on interfaces, collection framework and using libraries, e.g. for game-tree search, while for the beginner’s course wrappers are available that allow to simply access the board as plain integer array and thus fully concentrate on the core algorithm and the appropriate control structures. More advanced CS courses may find the PrimeGame being an interesting environment to study complexity issues or to focus on testing as both aspects can become critical when submitting a player to the tournament: Exceeding the given time limit per move or raising an exception while making a move will automatically disqualify the player.

Moreover, besides being a game for individual players, the PrimeGame can also be used very nicely to study the design and communication of cooperating agents: Instead of trying hard to implement the very best strategy, a group of students could try a different approach to win the tournament by "joining forces" and enabling their players to cooperate in a way that all but one of them support the remaining player. This can be done e.g. by applying a very poor strategy like the "Anxious Strategy" when it comes to play against a "friend" (the player to be supported) and to use a reasonably good one like the "Best Advantage Strategy" in all other cases. Here the focus will be more on defining an appropriate communication protocol as the only (legal) communication channel is the current set of numbers on the board. Studies have shown that normally a small number of 3 or 4 supporting players is sufficient to outperform even the best available player provided that the ranking table is calculated based on the total sum of points earned from all matches performed.

Thus, the Prime Game may also well be used to study various aspects far beyond a single specific topic ranging from programming using pre-defined interfaces, test-driven development, combinatorial algorithms and complexity issues, communication protocol design to cooperative agents to name just a few.

6 PAST EXPERIENCES

Having used the PrimeGame as a teaching and learning tool in under-graduate CS courses for over five years, we have collected valuable experiences in different contexts. Although a through quantitative analysis of its impact is yet to be carried out, we have found some interesting qualitative results based on observations, project assessments, and feedback. Apart from the effectively measurable performance of the player implementations across different courses and student groups, we believe that student motivation is significantly determined by the game presentation. This assumption is based on project outcomes in comparison to the initial console-based to the current GUI front-end.

Unlike our expectations, however, differences in student participation, performance, and consequently player outcomes were observed between German and Namibian students. To some extend, this may result from underlying discrepancies in primary and secondary education as well as curricula content and delivery modes in these countries. Yet another important factor, the cultural context, is similarly likely to having contributed to the slightly weaker average performance of Namibian students: "*students don't seem to enjoy given tasks regardless of their 'packaging'. Neither competition nor considered-fun activities resulted in higher marks or increased interest in the subject area.*" [3].

6.1 Experiences in Germany

The PrimeGame was first used in 2003 at the University of Applied Sciences Gelsenkirchen (FH) as part of the practical classes for the Introductory Course in Computer Science. Although the quality and performance of the student's players did not contribute to their semester marks at all, most students took part in the competition by submitting player implementations and were very ambitious to compete with their classmates and win the tournament. Interestingly, until very recently the best performing player developed by a German student dates back to the year 2003. Only in 2009, a few player implementations appeared which for most board sizes show better performance.

Since 2008, we also used the PrimeGame in the introductory course in CS of the IT Professional Programme at the IT-Center Dortmund, now belonging to the International School of Management (ISM). Here students can collect bonus points towards their semester mark by submitting a player and being ranked as one of the top-3 best performing players. In 2009, this led to a 97% submission rate (see Fig. 4). But success of the PrimeGame was not only in quantity but also as regards the quality of the player implementations: not only one but quite a few of them outperform the long-time best German player implementation from 2003. In average, players from the 2009 ITC PrimeGame Competition performed 48% better than the "Prime Number Strategy" used as the reference mark in Fig. 5.

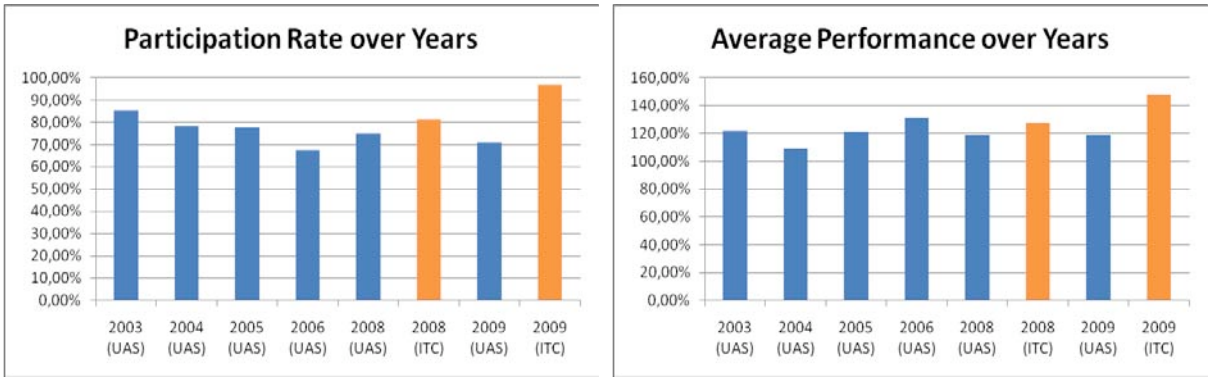

Fig. 4 & 5: Participation Rate and Average Performance over Years

Overall, independent the institution or course (UAS or ITC), when in practical classes it comes to introducing the PrimeGame and starting a two weeks period for player design, implementation and testing, most of the students almost instantly become highly motivated and keen to compete with classmates. As usual, however, some students still choose not to participate as the PrimeGame is not a mandatory assessment due to formal reasons. Compared to other practical exercises, this number has been observed to be much smaller. Thus, the PrimeGame works very well for improving student's involvement and participation.

6.2 Experiences in Namibia

Contrary to some of the experiences in German universities, the achieved outcomes from previous PrimeGame projects at the Polytechnic of Namibia in southern Africa did not yield as promising results. It should be noted that the Namibian context is quite different from the German one in that it has profoundly different cultural settings, as well as an IT market often absorbing students regardless of their academic achievements.

It has been witnessed in the past that the time spent by students on IT assessments in general, and programming projects in particular, is often limited to the absolute minimum. Thus OOP projects were often of poor quality, resulting in low pass marks a quite a number of students having to repeat the course.

We therefore introduced the PrimeGame as one of five programming projects within an Object-Oriented Programming (OOP) course in 2007 for the first time in Namibia. OOP is a third-semester course, following two prerequisite subjects, namely Introduction to Algorithm Design (IAD) and Introduction to Programming (IPR). Through these courses students are expected to having develop a theoretical understanding of fundamental algorithms as well as their underlying data structures, and to have gained hands-on practical programming experience.

While in the German setup the students' players were only used to determine bonus points for their class mark, we decided to try and create some additional incentives for the students to put extra effort into the PrimeGame player development. Similar to the German setup we incorporated the PrimeGame as one of five OO programming projects into the course. Furthermore we organised a large-scale public tournament, invited sponsors to provide bursaries, internships, IT equipment and smaller prices to the students with the most successful players, and eventually even had TV, radio and newspaper reporters present during the final event. While the public tournament (PrimeGame Namibian Open) has taken place only twice so far – in 2007 and in 2009 – the PrimeGame project has continuously been an integral part of the OOP course since its introduction in 2007.

Despite the fact that students seemed to have greatly enjoyed the PrimeGame events (regardless of an in-class or a public environment), the quality of the player submissions did not meet our expectations. In fact, the results were similar to other - more traditional - programming projects held in previous semesters, in that approximately 50% of the students did not receive a pass mark. Only about 5% of the students submitted well written players based on non-trivial algorithms. However, it is in fact usual for this number of students to hand in high-quality submissions regardless of the specific project's problem. We can thus deduct that the introduction of the PrimeGame did not have any significant effect on the motivation of students or on the quality of results.

Although the PrimeGame is definitely a valuable addition to the OOP course and can provide some fun during the competition events, we could so far not witness an impact on outcomes we were initially aiming at.

In the near future we are planning to use different aspects of the PrimeGame in other IT courses in an attempt to identify its full potential in the Namibian context. A more theoretic investigation into possible player algorithms, their applicability and complexity is envisaged for our Algorithm Design course in the first semester 2010.

7 FUTURE WORK

7.1 The PrimeGame in Computer Science Education

Although the main aspect of the PrimeGame in CS education is definitely the implementation of "intelligent" autonomous players, the game framework, rule-sets, font-end components and rendering modules among others present a variety of opportunities for many fields and a variety of assessment types in IT studies. This also includes game modding, thus letting students alter the engine or rules themselves, as similarly described by [1] for a more elaborated project. In the following paragraphs we are briefly introducing some of the potential areas of the PrimeGame which we believe can be successfully exploited in the future.

Player Algorithms

While we are currently using the PrimeGame mainly in undergraduate courses for Algorithm Design and Programming, far more advanced Player strategies could be developed in post-graduate CS courses. This obviously includes algorithms in artificial intelligence, but other areas such as Distributed Computing for the development of e.g. cooperative players also seem promising. Strategies based on large look-up tables or brute-force computations could similarly well be developed in cluster environments. Similarly, such approaches could incorporate networking aspects, resource constraints, or redundancy measures. The more theoretical aspects of PrimeGame players could furthermore be discussed in math-based courses on numerical methods or number theory to name just a few. Similar findings have been made by Gestwicki [4]: "*Specifically, computer science students will be attracted to a […] course on game-programming, and the […] course can be designed to teach concepts reaching far beyond the scope of the games themselves.*"

Game Framework

We are currently working towards a network-based PrimeGame environment. While we would like to continue using as much as the original GUI and Engine code from the current framework, new interfaces and service layers between those components will be introduced to allow for distributed competitions over TCP/IP networks. This project will be part of this year's Internet Programming course at the Polytechnic of Namibia. As we would like to utilize the HTTP protocol with full proxy support (in order to tunnel the traffic through firewalls), this project is not limited to component re-engineering, but will also involve high-level network protocol design and customization.

7.2 Teamwork and Intercultural Aspects

Having run PrimeGame competitions in two highly diverse environments - namely Germany and Namibia - already, it is planned to further expand the network of universities participating in PrimeGame competitions. While players from different places can easily compete in a common tournament, it might also be worthwhile to investigate intercultural teamwork aspects in distributed development projects. Based on our initial idea of providing the students with an interesting small-scale project early in their curricula, we believe this thought can very well be expanded to small-group projects in heterogeneous and spatially separated environments. Keeping such distributed project teams small, students might be able to more easily grasp group dynamics, and develop essential team skills in modern workgroup settings.

8 CONCLUSION

In this paper we have introduced the PrimeGame as a new tool for game-based learning in undergraduate CS programs. We have further outlined the game concept and motivated its use as a teaching tool in the first semesters, capable of successfully combining topics from a heterogeneous set of courses.

We have also presented the PrimeGame framework as it is currently being used for public competitions, and will be expanded in order to meet the envisaged course outcomes for under-graduate and post-graduate courses. We thus expect the PrimeGame to be a highly useful tool throughout CS programs, potentially covering a wide variety of not only individual but also group projects.

Necessary prerequisites in terms of prior student knowledge have been given, before having outlined our intended aims in employing the PrimeGame as a teaching and learning tool for applied programming.

Using the PrimeGame throughout several years, including large-scale public competitions in Namibia, we have also done a qualitative comparison of its impact on student learning results in selected introductory programming courses. The evaluation of the game in German and Namibian universities has shown significant differences in this regard. In the German context – which is likely to be stereotypical for most western ones – the game has been found to provide not only a highly effective, but also a very entertaining project for individual student work. Yet such findings could not fully be confirmed in the Namibian context.

However, for both environments we could identify a positive impact on the practical application of a *combination* of diverse skills usually taught in disjoint courses. Given the task of a PrimeGame player development, students seemed to more easily appreciate previously “dry” – and often decoupled – topics such as complexity analysis, data structure design, or unit testing.

We assume the above mentioned benefits to be largely due to a highly limited – and thus easily graspable and manageable – project scope in conjunction with an entertaining project. The witnessed differences in the perceived “fun factor”, and thus in student motivation, are assumed to result from underlying diversities in the competitive aspects of culture.

We have also presented a number of future opportunities around the PrimeGame concept and framework, which we hope can help us carrying over the previous success in few courses with increased student motivation and quality output into a greater variety of courses and projects. In terms of our failures to fully meet the intended objectives in Namibia, future studies to verify our assumptions and to propose promising alternatives are yet to be conducted. We will also be looking into additional opportunities arising from our experiences with the PrimeGame in order to employ selected aspects of this (or similar) game(s) in diverse international environments.

Finally, we would like to point out the PrimeGame's potential as a collaboration tool among CS education institutions world-wide. Through involving lecturers and students at participating institutions simultaneously, and by allowing a public audience to follow entertaining events, we believe this game is a perfect vehicle to foster individual, institutional and professional partnerships. It is thus our hope that we can create further interest to participate in international PrimeGame competitions in the near future.

9 ACKNOWLEDGEMENTS

We would like to thank our institutions, the University of Applied Sciences Gelsenkirchen, the IT-Center Dortmund GmbH and the Polytechnic of Namibia, for their support in hosting the PrimeGame.

We would further like to thank our students who have participated in the PrimeGame, and who not only came up with with a number of highly efficient and well-performing player algorithms, but also initiated very interesting discussions around different aspects of the game.

Finally our thanks go to all previous and future sponsors of the PrimeGame, as well as to the countless individuals who have contributed great amounts of time and effort towards the success of the PrimeGame competitions. Without their support, generosity and initiatives we would not have had such successful and entertaining events.

References

[1] El-Nasr, M. S. & Smith, B. K. (2006). Learning through game modding. Comput. Entertain., 4 (1), 7.

[2] Fachhochschule Gelsenkirchen. Modulhandbuch (Syllabus) Bachelor of Science in Business Computing, http://www2.fh-gelsenkirchen.de/FH-Sites/fachbereiche/fileadmin/fb7/Downloads/Pruefungs-_und_Studienordnung/Modulhandbuch_BA_Wirtschaftsinformatik_09-07-07.pdf Last Accessed: 4 February 2010

[3] Fendler, J. & Winschiers-Theophilus, H. (2010). Towards contextualised software engineering education: An african perspective. To be published. ICSE2010, Cape Town, South Africa, 2010

[4] Gestwicki, P. & Sun, F.-S. (2007). On games, patterns, and design. In SoD '07: Proceedings of the 2007 Symposium on Science of Design, pp. 17-18, New York, NY, USA. ACM.

[5] IT-Center Dortmund. IT Professional. http://www.itc-dortmund.de/it_professional_studieninhalt.php Last Accessed: 4 February 2010

[6] Muratet, M., Torguet, P., Jessel, J.-P., & Viallet, F. (2009). Towards a serious game to help students learn computer programming. Int. J. Comput. Games Technol., 2009, 1-12.

[7] Office of the Registar (2010). "Prospectus for Undergraduate Studies". Polytechnic of Namibia, Windhoek, Namibia

[8] Rankin, Y., Gooch, A., & Gooch B. (2008): The Impact of Game Design on Students' Interest in CS. In Proc. GDCSE Conference 2008. Cozumel, Mexico.

[9] Wang, A. I. & Wu, B. (2009). An application of a game development framework in higher education. Int. J. Comput. Games Technol., 2009, 1--12