Christian Reuter, Stefan Göbel, Ralf Steinmetz:
Approximating Balance in collaborative Multiplayer Games. Robin Munkvold and Line Kolås: 9th European Conference on Games Based Learning, p. 449–455, Academic Conferences Limited, October 2015. ISBN 978-1-910810-58-3.

Approximating Balance in collaborative Multiplayer Games

Christian Reuter, Stefan Göbel, Ralf Steinmetz
TU Darmstadt, Darmstadt, Germany
christian.reuter@kom.tu-darmstadt.de
stefan.goebel@kom.tu-darmstadt.de
ralf.steinmetz@kom.tu-darmstadt.de

Abstract: Multiplayer games allow players to play together, allowing them to train their socials skills and enabling collaborative learning in educational settings. But while balancing a game's challenge is a well-researched topic, to the best of our knowledge there are no definitions and algorithms taking into account the individual contribution of each player in a collaborative setting. Unequal contributions however can lead to one player completing the game alone while the others are not playing at all in the worst case.

In this paper we provide a novel definition for collaborative balancing suited for multiplayer games with fixed and free roles, including metrics for heuristically calculating effort, waiting times and options available for each player. These metrics have also been implemented in a graph-based analysis algorithm, which has been applied to an example game in order to show the feasibility of this approach.

Keywords/Key Phrases: multiplayer, collaborative, cooperative, balancing

1. Introduction

It has been shown that well-designed multiplayer games can enable collaborative learning while also training the players' soft skills. When creating a game for a group of learners however, it is quite hard to guarantee that every group member will be exposed to the same amount of learning content and that they contribute the same amount towards solving the game. If this is not the case, multiple problems arise that negatively impact both the learning outcome as well as the players' motivation.

For example, players with pre-existing knowledge might be able to solve an unbalanced game on their own while the others do not take part in the learning process and thus do not receive the intended knowledge. The better players in turn could complain they had to solve all the tasks on their own or, depending on the game design, that they had to wait for the others constantly. This can be viewed as a balancing problem since the game's tasks must be distributed between the individual players in a fair manner.

This paper is structured as follows: First we discuss the related work on game balancing in Chapter 2. After that we provide a novel definition for collaborative balancing in Chapter 3 and corresponding metrics in Chapter 4. We then describe the prototypical implementation of these metrics in an authoring tool in Chapter 5, discuss its results when applied to an example game in Chapter 6 and conclude with a short summary in Chapter 7.

2. Related Work

Although there is a general consensus that balanced games are fair towards their players, suggestions on which properties a game should have to achieve that differ widely.

Balance can be viewed on multiple levels. The first one is the game's design, which is independent of concrete players. For Sirlin a balanced game design provides equal chances of winning to similarly skilled players and presents them a large number of viable options during play. The utility value of these options can differ depending on the current situation, as long as there are no strategies that are always right or wrong (Sirlin 2002). Newheiser provides a similar definition, but notes a contradiction in multiplayer games: New players want to be able to win against experienced ones, but experienced players want win consistently due to their higher skill level. Therefore, he suggests a mixture between random and skill elements (Newheiser 2009). Adams in contrast states that the leading player should change frequently to make games more interesting, but the better player should always win after some time (Adams 2002).

To achieve a balancing between different options, Leigh et al. propose a coevolutionary algorithm that automatically develops strategies and then modifies the game in such a way that there is no dominating one. The drawback is that the space for potential strategies is often too large and

therefore not every strategy can be tested (Leigh et al. 2008). Finding optimal solutions algorithmically has even been argued to be NP-hard (Chen et al. 2014).

Another level on which a game can be balanced is its runtime behaviour, where it might adapt itself to better suit differently skilled players. Andrade et al. used reinforcement learning to unnoticeably keep a singleplayer game balanced (Andrade et al. 2005). When multiple players play against each other it is also an option to quietly support the weaker player, which has been done in a shooting game (Bateman et al. 2011) and a racing game (Cechanowicz et al. 2014). It must be noted however that depending on the individual player's attitudes they might perceive such aids as unfair. Instead it might be more beneficial to match them against appropriate opponents instead (Delalleau et al. 2012).

Other sources of unfairness in a multiplayer setting are external factors such as latency issues (Zander et al. 2005) and players using illegitimate methods to gain an advantage (Yeung et al. 2006)

It must be noted that all of these approaches take a competitive view, i.e. the player playing against the game in a singleplayer setting or different players playing against each other in a multiplayer setting. Nevertheless, they could be applied in a collaborative setting for balancing the players as a group against the game. However, to the best of our knowledge there are no definitions or metrics that take the interplay between the collaborating players into account. This is especially important since it has been shown that in working environments effectiveness and the amount of social interactions increases when workers feel like they are treated fairly (Whitman et al. 2012), which is also relevant in collaborative learning settings.

3. Definition: Collaborative Balancing

In order to balance collaborative games where players work together to achieve a common goal, e.g. reaching the end of the game, a new definition including the interplay inside teams of any number of players is needed.

When talking about fairness in a team most people are interested in individual contributions. It is perceived as unfair if only a part of the group is doing the actual work while others profit from their effort, which is also known as free-riding (Strijbos & De Laat 2010). This is especially problematic in a collaborative learning setting, where participation means being exposed to the learning content. Therefore, players that do not actively take part might miss some of the learning content. It is easy to see that all players should contribute roughly the same amount of effort. Waiting times are directly related to the effort other players provide and should be balanced as well.

It has been argued in competitive definitions that it is also beneficial to provide a variety of interesting decisions. In a collaborative setting this must not only be true for the group as a whole, but for each individual player. A player role that forces a player to repeat a single action over the course of the whole game will not be very interesting, even if the group as a whole can choose different paths through the game. Similarly to effort, options should be balanced between the players.

We therefore provide the following definition for a collaborative balancing:
A collaborative or cooperative game is considered balanced between group members when the individual efforts and waiting times required to solve the game as well as the number of options a player has along the way are equally high and uniformly distributed throughout the game.

4. Approximating Collaborative Balance

In order to measure how balanced a game is the abstract concepts of effort, waiting times and possibilities must be mapped to actual game elements. Since players interact with games using actions like clicking a button or pressing a key, these actions can be used for approximating the balance (with an additional weighting to model their difficulty or complexity).

These actions can be available for any player or for specific players / roles only. If they are available for multiple roles it must be decided during play which player should perform those tasks. This places some responsibility on the players, which is more interesting for them and gives them room to train their social skills, e.g. by discussing different alternatives. The drawback of freely available actions is that they cannot be attributed to a single player during design and so the balance may vary based on how the game is played later on.

Most games are also non-linear so that players can take different paths to the game's ending, either by choosing between multiple options or by taking optional detours. To accurately measure the balance it would therefore be necessary to count the player actions on every path. As some paths are more obvious they should also be weighted based on the probability with which they are chosen. This probability however is not known and in most cases an infinite number of paths through the game exist, for example when players are able to move back and forth through several rooms. But even without such loops the number of paths through most games is still too large to be checked exhaustively.

We therefore suggest a heuristic solution as a trade-off between accuracy and runtime. This heuristic uses the state space graph of the game, which contains all possible game states as nodes and ways for the players to trigger transitions between them. It then calculates all possible end states e, which does not only include different game endings, but also different ways to reach a single ending resulting in differently set game variables. Then for every player p_m the path to each ending that minimizes a certain metric (e.g. effort) for this player $minPath(e, p_m)$ can be calculated to get a lower bound of that metric, so the player must contribute at least this effort for the group to reach this ending. It is necessary to calculate those paths for each player independently since the minimal path might be different for each player.

Maximizing instead of minimizing is not feasible in most cases since loops could lead to infinite maximum paths. If the state space is not available it is also possible to calculate the solutions for the game with appropriate AI algorithms and to use these as paths instead. If the number of possible paths is still too high a random sample can be used instead.

After the minimal paths $minPath(e, p_m)$ of each player p_m to each ending state e are known the effort for all player p on this path $E^{p}_{p_m e}$ can be calculated, which yields the effort each player p has to contribute in order to minimize the effort of p_m. This effort is defined by the combined effort of all actions on the path that only one particular player p can trigger. Actions that can be triggered by multiple players do not count towards the minimal effort of a certain player, as no player is forced to perform them.

$$E^{p}_{p_m e} = \sum_{a \in minPath(e, p_m)} effort(a, p) \qquad effort(a, p) = \begin{cases} weight(a), & only\ p\ can\ trigger\ a \\ 0, & else \end{cases}$$

The combined effort of all actions on this path $E^{all}_{p_m e}$ is the sum of all efforts on this path. Since our model supports actions that can be triggered by multiple players it differentiates further between this sum and the sum of all actions that can only be triggered by a single player ($E^{any}_{p_m e}$).

$$E^{all}_{p_m e} = \sum_{a \in minPath(e, p_m)} weight(a) \qquad E^{any}_{p_m e} = \sum_{i=0}^{n} E^{i}_{p_m e}$$

In a game where every action is assigned to a single player these two values are the same and the effort can be calculated accurately. But if many actions can be triggered by multiple players these values differ and the approximated balance can be changed greatly by the players themselves depending on who execute these flexibly assigned actions. The accuracy of the calculated effort $A^{E}_{p_m e}$ can therefore be calculated by comparing these two values to each other.

As different paths through this game might require a different overall effort one should not compare the minimal effort of each player on different paths, but the percentage this player has to contribute compared to the others ($E^{p\%}_{p_m e}$). On each ending e these should be similar for each player.

$$A^{E}_{p_m e} = \frac{E^{any}_{p_m e}}{E^{all}_{p_m e}} \qquad E^{p\%}_{p_m e} = \frac{E^{p}_{p_m e}}{E^{any}_{p_m e}}$$

While the effort is defined by the actions a certain player must take, the waiting time is the effort a player is not able to participate in as he or she must wait for someone else to complete them. An action that multiple players are able to perform is therefore neither effort nor waiting time for any of

them, as they can execute it but are not forced to do so. As such, the formulas for effort and waiting time are similar with one important distinction: The waiting time percentage is based on the value of all actions, not on those that can be uniquely assigned to one player as everyone else is waiting when one player acts. Therefore, there is also no accuracy for this value.

$$W^{p}_{p_m,e} = \sum_{a \in minPath\,(e,p_m)} waiting\,(a,p) \qquad waiting\,(a,p) = \begin{cases} 0, & p\ can\ trigger\ a \\ weight\,(a), & else \end{cases}$$

$$W^{all}_{p_m,e} = \sum_{a \in minPath\,(e,p_m)} weight\,(a) \qquad W^{p\%}_{p_m,e} = \frac{W^{p}_{p_m,e}}{W^{all}_{p_m,e}}$$

This approximation only counts actions which are required for solving the game (i.e. they lay on the minimal path to an end state). In some cases waiting players could perform optional actions or even trigger a required action in parallel, which is not represented in this calculation.

The options each path offers to the players can be calculated like the waiting time, using the possible actions for each player after each step as the metric.

$$O^{p}_{p_m,e} = \sum_{a \in minPath\,(e,p_m)} options(a,p) \quad options(a,p) = \sum_{o \in successors(a)\ |\ p\ can\ trigger\ a} weight\,(o)$$

$$O^{all}_{p_m,e} = \sum_{o \in successors\,(a)} weight(o) \qquad O^{p\%}_{p_m,e} = \frac{O^{p}_{p_m,e}}{O^{all}_{p_m,e}}$$

Instead of calculating individual paths for waiting times and options, their calculation could also be based on the effort optimizing path. This trades accuracy – there might be a path with a lower metric – for a greatly reduced runtime as fewer path must be calculated.

After the percentages of each metric for every player and path are calculated they must be aggregated to get the overall picture. As the players' values for a specific ending should be similar their standard deviation over all paths should be minimal, which can be used as an aggregation metric. It must be noted however that although the mean percentages lie in the interval $[0,1]$, the worst case standard deviation SD_{worst} (one player does all the effort instead of the optimal $\frac{1}{n}$ distribution) is dependent of the number of players n. In order to compare the standard deviations of any metric x over different games they should therefore be normalized in relation to SD_{worst}.

$$SD_{worst} = \sqrt{\frac{\left(1-\frac{1}{n}\right)^2 + (n-1) * \left(0-\frac{1}{n}\right)^2}{n}} = \frac{\sqrt{(n-1)}}{n} \qquad SD_{norm}\,(x) = \frac{SD(x)}{SD_{worst}}$$

Finally, the normalized standard deviations for effort, waiting time and options as well as the accuracy for each ending state must be interpreted: If the average and maximum of the standard deviations is low the game is balanced (the metrics are similar for all players). If only the average is small most paths are balanced except a few outliers and if it is high most paths are unbalanced.

The accuracies average and minimum over all paths describes how much the result can be modified by players executing actions that are also available to others. Averaging the absolute effort, waiting time and possibilities for each player instead of looking at the standard deviation can show which players are favored by an unbalanced game. These values can be similar however, even if the standard deviation is high. This means that each path through the game favors different players, so advantages equalize themselves over all paths. Since few players play the game many times using different paths, this is still a balancing problem.

If the analysis shows anomalous values the algorithm highlights problematic paths as sequences of player actions, which allows authors to modify them, for example by assigning actions to other players. Since all of these metrics can be calculated during the design / authoring phase, the balancing can be approximated without a working game or playtesters.

The overall algorithmic complexity of this approach is defined by the path finding algorithm, which must be run for each combination of player and end state.

5. Implementation

We implemented this approach for approximate balancing calculation in our authoring tool StoryTec (Mehm et al. 2009), with which authors can create scene-based single- and multiplayer games. Therefore, the balancing calculation can be performed on any game created with the tool, although it can only show the overall effort for singleplayer games.

The state space calculation is provided by an existing algorithm for transforming games into petri net models in order to detecting structural errors (Reuter et al. 2015). The authoring tool then uses the well-known Dijkstra's algorithm for calculating the shortest path to each end state. It's faster A* extension which uses a heuristic to guide the search in promising directions first can also be used if there is a known lower bound for the costs from each node to the end. A valid bound would be the number of actions to the end goal, which can be obtained with a breadth-first-search on the graph starting from the end node. This calculation must be repeated for each end node, but can be re-used for every player. Since this operation also takes time, it is more beneficial to use A* for a larger number of players and for state spaces where it performs especially well compared to Dijkstra, i.e. in graphs that expand in multiple directions. Therefore, both options were implemented.

For approximating the weight of individual actions multiple heuristics were developed, based on the principle of ActionSets used in the authoring tool. An ActionSet is a tree of game (re-)actions to a user input stimulus with branches based on variable conditions. The first heuristic just counts the user inputs and treats every ActionSet the same, assigning it a value of 1. The second one emphasizes the overall complexity of the tree, counting the actions it contains. Depending on the structure of the tree and the variables used in its conditions, only some of the actions might be executed in a given game state. To respect that the third heuristic counting the actually executed actions that change the game state (optional options like requesting hints are ignored). And last but not least authors can override each heuristic by manually setting the difficulty of ActionSets according to their expert knowledge.

In order to find the minimal value of each metric for a player a naïve approach could override all transitions that could be triggered by other players with a weight of zero, as they are not relevant for the minimum. However this would result in a greatly increased runtime, as the pathfinding algorithm would search all of these transitions first before triggering one by the player in focus (which it must do eventually if the action is required to solve the game). In order to not explore the whole state space it is therefore advisable to choose weights of at least one for those irrelevant transitions. As actions of other players must be favoured in order to find the minimal effort for the player in focus there must be a numerical advantage of those actions, so we propose to multiply the weights of these transitions with 20 instead. This means that if all player actions are judged equally complex the pathfinding algorithm would search 20 consecutive transitions done by other players before eventually considering one for the player whose effort should be minimized. This number can be tuned as a trade-off between searching the whole state space and correctness: If this player's action could be circumvented with 21 other actions the path would not be optimal and the calculated metric would be too high. But since the calculations are approximations anyway such an error is not critical.

If A* is used for pathfinding an additional heuristic for guiding the search is required, adding further constraints to the action weighting. The A*-heuristic must be admissible, i.e. its values must not be larger than the actual distance to the end state. Since the minimal number of steps towards the goal is used as this heuristic, the weight of every step cannot be smaller than one.

When calculating the final metrics for each path it is not necessary to take pathfinding costs into account, so the unmodified weights can be used.

6. Results

The implementation was then tested by analysing a well-received multiplayer adventure game (Reuter et al. 2012). The game is designed for two players and uses relatively strictly enforced roles, with most actions being assigned to one player only. Our user study had shown that players of both roles felt involved during the majority of the game, which means that according to our definition it was balanced for them.

The state space of a large subsection of this game spanning thirteen minutes of playtime has already been calculated in our previous work (Reuter et al. 2015). Despite having a single ending the game's state space contains 88 different ending states. We then applied our balancing metrics in various parameter combinations (Table 1) using different weight heuristics (input, complexity and actual actions), path calculations (optimizing effort only or each metric separately) and path finding algorithms (Dijkstra and A*).

Table 1: Evaluation results

			Time		Accuracy		Effort SD		P1	P2	Waiting SD		Options SD	
Weight	**Optim.**	**Pathf.**	**Imp.**	**Calc.**	**Min**	**Avg**	**Avg**	**Max**	**Avg**	**Avg**	**Avg**	**Max**	**Avg**	**Max**
Input	Effort	Dijkstra	2 s	7 s	0,92	0,99	0,08	0,17	20	20	0,08	0,17	0,02	0,07
Input	Effort	A*	2 s	4 s	0,92	0,99	0,08	0,17	20	20	0,08	0,17	0,03	0,07
Input	Sep.	Dijkstra	2 s	20 s	0,92	0,99	0,08	0,17	20	20	0,08	0,17	0,04	0,08
Input	Sep.	A*	2 s	11 s	0,92	0,99	0,08	0,17	20	20	0,08	0,17	0,04	0,08
Compl.	Effort	Dijkstra	2 s	10 s	0,75	0,97	0,11	0,25	90	97	0,13	0,26	0,13	0,18
Compl.	Effort	A*	2 s	7 s	0,75	0,97	0,11	0,25	90	97	0,13	0,26	0,13	0,18
Compl.	Sep.	Dijkstra	2 s	45 s	0,75	0,97	0,11	0,25	90	97	0,11	0,25	0,07	0,13
Compl.	Sep.	A*	2 s	25 s	0,75	0,97	0,11	0,25	90	97	0,11	0,25	0,07	0,13
Actual	Effort	Dijkstra	2 s	6 s	0,94	1,00	0,11	0,22	26	30	0,11	0,22	0,32	0,47
Actual	Effort	A*	2 s	4 s	0,94	1,00	0,11	0,22	26	30	0,11	0,22	0,32	0,47
Actual	Sep.	Dijkstra	2 s	24 s	0,94	1,00	0,11	0,22	26	30	0,11	0,22	0,16	0,33
Actual	Sep.	A*	2 s	13 s	0,94	1,00	0,11	0,22	26	30	0,11	0,22	0,16	0,33

The weight heuristic is most influential as it governs the impact of each individual player actions on the overall result. As complexity results in the largest value range of individual actions it also results in a larger variance in balancing, i.e. an average effort balance of 0.11 (worst case 0.25). This is due to single complex action skewing the result towards a single player. Using actual effects and input in contrast suggests that the game is more balanced, i.e. 0.11 (worst case 0.22) and 0.08 (worst case 0.17). Therefore authors must select a heuristic that is most appropriate for the game under test over even annotate actions using their own weights. As the overall value range is between zero and one (one indicating completely unbalanced games), all of these values however suggest a relatively balanced game.

When investigating the absolute effort player two has to provide 1% to 16% more effort over all paths, so the slight unbalance could be improved by shifting some actions to player one. The accuracy of these results is greater than 97% on average with a worst case of 75% on individual paths, meaning that there are only few actions that the players can choose to distribute themselves in order to influence the balancing. As the ActionSets behind those actions are relatively complex compared to the other ones their influence is most obvious when using this metric.

Waiting times are also pretty balanced, with an average standard deviation of up to 0.13 (worst case 0.26). The possibilities produce reasonable values as well, 0.16 on average, but the worst case of 0.33 suggests that there are some paths through the game that do not balance the possibilities well. It is also interesting to note that this value gets much worse (0.32 instead of 0.15 on average) when there is no separate path calculated for the possibilities, as the effort path minimizes a completely different metric. The waiting times in contrast do not vary much when calculated on an effort path, as the metrics are similar to each other.

In regards to computation time the initial setup including state space import takes about two seconds, the actual balancing calculations vary between ten and twenty seconds when only effort paths are calculated. As expected, this time is nearly tripled when the paths are also calculated for the other two metrics. Since our results already showed that effort and waiting time are quite similar, a better

approach would be to only calculate effort and possibilities, doubling the time. The runtime can be decreased further by using the A*-Algorithm instead of Dijkstra, which took 25% to 50% less time for our example game while yielding the same result. Nevertheless, all of these calculations can be done in almost real-time and therefore can be calculated quite often during development.

7. Conclusion

In this paper we proposed a balancing definition explicitly addressing collaborative games where multiple players work towards the same goals. This definition includes the effort required by each individual player to solve the game, their waiting time and the possible actions the game provides for them. We then developed an approach and formulas to approximately calculate the balancing of any given game and implemented this concept into an authoring tool for scene-based games. Using this implementation we showed that an example game that was described as generally balanced could indeed reach good values and that the calculations could be done in a reasonable time span of less than a minute.

A current limit of this work is that while the general definition and concept can be used for any collaborative game, the implementation and results focus on scene-based games. As the state space of games from other genres is comparatively large it is not always possible to exhaustively calculate every path through them. It is therefore necessary to implement alternative path calculations in order to apply our approximations to other games as well, which will be part of our future work.

Other future work will include not only pointing out imbalances, but extending our algorithms to also provide suggestions on how to fix anomalies. While being straightforward in unambiguous cases where every path through the game favours the same player, modifying actions in cases where only some paths are unbalanced might impact formerly balanced paths as well. Suggesting changes therefore constitutes a separate optimization problem. And aside from balancing the game during its creation, future work could also balance the game during runtime for differently skilled players by re-assigning responsibilities on the fly.

References

Adams, E., 2002. Balancing Games with Positive Feedback. *Gamasutra*, p.1. Available at: http://www.designersnotebook.com/Columns/043_Positive_Feedback/043_positive_feedback.htm [Accessed April 24, 2014].

Andrade, G. et al., 2005. Automatic Computer Game Balancing: A Reinforcement Learning Approach. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*. AAMAS '05. New York, NY, USA: ACM, pp. 1111–1112.

Bateman, S. et al., 2011. Target assistance for subtly balancing competitive play. *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, p.2355.

Cechanowicz, J.E. et al., 2014. Improving Player Balancing in Racing Games. In *Proceedings of the First ACM SIGCHI Annual Symposium on Computer-human Interaction in Play*. CHI PLAY '14. New York, NY, USA: ACM, pp. 47–56.

Chen, H., Mori, Y. & Matsuba, I., 2014. Solving the balance problem of massively multiplayer online role-playing games using coevolutionary programming. *Applied Soft Computing*, 18(0), pp.1–11.

Delalleau, O. et al., 2012. Beyond Skill Rating: Advanced Matchmaking in Ghost Recon Online. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(3), pp.167–177.

Leigh, R., Schonfeld, J. & Louis, S.J., 2008. Using coevolution to understand and validate game balance in continuous games. *Proceedings of the 10th annual conference on Genetic and evolutionary computation - GECCO '08*, p.1563.

Mehm, F. et al., 2009. Authoring Environment for Story-based Digital Educational Games. In Y. Cao et al., eds. *Proceedings of the 1st International Open Workshop on Intelligent Personalization and Adaptation in Digital Educational Games*. CEUR Workshop Proceedings, pp. 113–124.

Newheiser, M., 2009. Playing Fair: A Look at Competition in Gaming. *Strange Horizons*, p.1. Available at: http://www.strangehorizons.com/2009/20090309/newheiser-a.shtml [Accessed April 24, 2015].

Reuter, C. et al., 2012. Multiplayer Adventures for Collaborative Learning With Serious Games. In P. Felicia, ed. *6th European Conference on Games Based Learning*. Reading, UK: Academic Conferences Limited, pp. 416–423.

Reuter, C., Göbel, S. & Steinmetz, R., 2015. Detecting structural errors in scene-based Multiplayer Games using automatically generated Petri Nets. In *Foundations of Digital Games 2015 (accepted for publication)*.

Sirlin, D., 2002. Balancing Multiplayer Games. *Sirlin on Game Design*, p.4. Available at: http://www.sirlin.net [Accessed April 24, 2015].
Strijbos, J.-W. & De Laat, M.F., 2010. Developing the role concept for computer-supported collaborative learning: An explorative synthesis. *Computers in Human Behavior*, 26(4), pp.495–505.
Whitman, D.S. et al., 2012. Fairness at the collective level: A meta-analytic examination of the consequences and boundary conditions of organizational justice climate. *Journal of Applied Psychology*, 97(4), pp.776–791.
Yeung, S.F., Lui, J.C.S. & Yan, J., 2006. Detecting cheaters for multiplayer games: theory, design and implementation[1]. *CCNC 2006. 2006 3rd IEEE Consumer Communications and Networking Conference, 2006.*, 2, pp.1178–1182.
Zander, S., Leeder, I. & Armitage, G., 2005. Achieving fairness in multiplayer network games through automated latency balancing. *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology - ACE '05*, pp.117–124.