Solving Motion and Action Planning for a Cooperative Agent Problem Using Geometry Friends

Ana Salta(✉), Rui Prada, and Francisco Melo

Instituto Superior Técnico and INESC-ID, Av. Prof. Doutor Aníbal Cavaco Silva, 2744-016 Porto Salvo, Portugal
{anasalta,rui.prada}@tecnico.ulisboa.pt, fmelo@inesc-id.pt

Abstract. In this paper we discuss the development of agents for the Geometry Friends game, which poses simultaneously problems of planning and motion control in an physics-based puzzle and platform 2D world. The game is used in a competition, held yearly, that challenges participants to solve single player and cooperative levels. Our work addresses the two. The approach followed uses Rapidly-Exploring Random Trees with strategies to accelerate the search. When comparing with other agents on the competition, our results show that our agents can solve the single player challenges without overspecialization and are also promising for the cooperative levels with either agent-agent and human-agent players.

Keywords: Artificial agent · Rapidly-Exploring Random Trees · Motion planning · Motion control · Replanning · Human-agent cooperation

1 Introduction

Intelligent agents and Multi-Agent Systems (MAS) are becoming more popular as they simplify tasks from our daily activities to the industry world, scientific research and so forth. Planning and execution are subjects that have been explored with great interest for they can be applied to single agents and MAS in industries like robotics and video games. One of the most challenging problems for planning and execution is human-agent cooperation. If the agents are not able to perform their tasks well enough, even those that can be done by themselves, then they cannot aspire to cooperate with a human. Simple virtual problems are created, simulated and tested so their solution can be presented through their abstraction and easily adapted to similar real life problems. Many of these virtual problems are used in competition scenarios to appeal researchers to participate, like Geometry Friends [8], a cooperative physics-based puzzle and platform game with a problem of real time cooperation as well as motion planning and control. Solving problems like Geometry Friends which study cases of real time planning and fine motor coordination can help develop solutions for real life situations such as search and rescue, in which there is a necessity of

P. Moura Oliveira et al. (Eds.): EPIA 2019, LNAI 11804, pp. 86–97, 2019.
https://doi.org/10.1007/978-3-030-30241-2_8

highly efficient and effective responses from the involved agents, whether human or artificial. The main objective is to develop a pair of agents that can cooperate with another player, human or artificial. In this work, we select one of the most promising agents of the Geometry Friends competition, the RRT agents, which uses the Rapidly-Exploring Random Trees (RRT) algorithm, and study how to improve them by testing the combination of already proposed strategies, and by adding other strategies to help the search and control.

Our main contributions consist on a set of strategies to accelerate the RRT search, considering the context of the problem it might be used on, as well as a controller to follow the plan which acknowledges failed actions and attempts to recover from them. We then present a set of agents that perform well during the single player levels, without overspecialization, and are able to solve simple cooperation levels. This base work can be used to study further methods of cooperation and help open new doors to a generalized approach with applications in the real and virtual worlds.

In Sect. 2, we present the background to contextualize our work and the related work is referenced in Sect. 3. Sections 4 and 5 respectively describe the development of the agents for the single player and multiplayer levels. We then test and discuss the results in Sect. 6 with the final conclusions and future work discussion at Sect. 7.

2 Background

2.1 Geometry Friends

Geometry Friends [8] is a cooperative physics-based puzzle and platform game that has single and multiplayer modes. To succeed in the game, the player, or players, must catch all the diamonds present in each level by simply touching them. Some levels focus on puzzle-like problems and other levels focus on motion control. Figure 1 shows the actions each character can perform and an example of a cooperative level where the purple rhombus are the diamonds to catch and the black areas are platforms, obstacles the characters cannot get through.

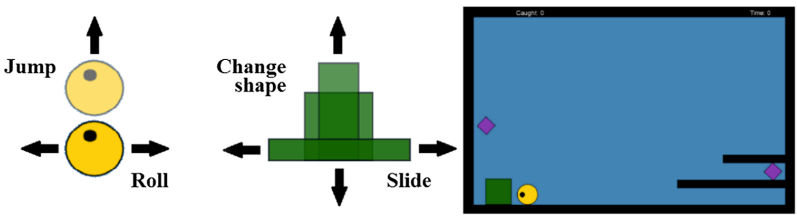

Fig. 1. Left: Characters possible actions. The circle can roll and jump while the rectangle can slide and change shape by morphing up or down. **Right:** Example of a cooperative level. To catch the diamond on the right, the rectangle must morph down and slide above the circle while next to the platform.

The main objective of this game is to study solutions for human-agent and agent-agent cooperation problems, from action planning to action coordination. To promote this, a competition is held each year where the proposed agents are tested with levels made public and others kept private.

2.2 Rapidly-Exploring Random Trees

The RRT algorithm was introduced by LaValle et al. [6] to solve the problems of other randomized approaches, like the non-uniform coverage of the state space and the planning not being suitable for nonholonomic or kinodynamic problems. A tree $\mathcal{T}$ is limited by K iterations representing the number of vertices of the tree. For each value of K, a random state x, outside $\mathcal{T}$, is selected for extension. The method of extension chooses the nearest state to the given state x, x_{near}, already in $\mathcal{T}$, with the help of a distance metric. With an action, u_{new}, that can be chosen randomly or by testing all possibilities, an attempt is made to move towards x, from x_{near}, by applying u_{new} to x_{near} for a period of time and verifying if the state x_{new} is generated. If x_{new} is the same as x, that state is considered reached, else it is considered as advanced. If the state x_{new} is not possible due to violations of some constraint, it is then considered as trapped. The goal of the algorithm is to search high-dimensional spaces with algebraic and differential constraints, by biasing the exploration to the unexplored state space and guide the search towards the randomly selected states [7].

3 Related Work

Some solutions have already been proposed for the single-player mode of Geometry Friends, including one that uses the RRT algorithm, the RRT agent [9]. The RRT agent shows promising results but is still not polished enough to solve some levels in a satisfactory way, mainly due to its controller, which fails to perform some actions of the plan, and the lack of replanning when the agent fails to do an action or the plan is invalidated by the other player.

Some strategies have already been suggested as an extension for the RRT algorithm. The Execution Extended RRT (ERRT) planner [2], for example, introduces replanning. The planner uses a waypoint cache of states from previous plans. When the RRT selects the next state, it is biased to choose a state from the cache instead of a random one from the free space, thus biasing the planner towards the previously successful plans. To avoid constant replanning in an environment that is constantly changing, the Variable Level-of-Detail (VLOD) strategy was proposed [13]. In these environments, it might be more efficient to execute the plan for a time interval and only then replan. When replanning, the search is focused on obtaining a path plan with short-term results while the future is seen in a more global way, but still giving a full start-goal path plan. To improve the efficiency of the RRT algorithm, strategies to bias the search towards the goal, like RRT-GoalBias, or towards an area around it, like RRT-GoalZoom, when selecting a state were also developed [7]. [12] suggested the use

of tactics and skills, for the action selection, which introduces the Behavioral Kinodynamic Balanced Growth Trees (BK-BGT), that avoids the use of a distance metric, which can be hard to define, when choosing the next state in the tree, by choosing one already on the tree using the ratio between the average of the leaves depth and the average of the branching factor to decide if a random leaf or non-leaf state is selected.

We also want our solution to be a start towards a solution that can work on Geometry Friends cooperation levels, either agent-agent or human-agent players. For this we need to understand, in a simple way, what important factors the human cooperation has. The human interaction is based on aspects that are extremely complex to bring to the context of artificial agents. People actions are affected by social factors [1,4,11], culture, values, and even psychological states that might make the same person display different behavior towards similar situations [5]. Bradshaw et al. [1] mention that human coordination is done with the help of signals or other types of messages, meaning communication is essential to interaction. Human-agent communication is very limited in the game, turning this into a greater challenge. Teamwork and trust are also important forms of human interaction that is essential for the evolution of our societies. Van Wissen et al. [10] show that people do have different reactions and ways of playing when they believe that they are playing with humans or agents. When cooperating, an agent must be able to adapt to the human in order to achieve the common goals. For that, the agent needs to change its plan dynamically to follow the human's actions [4].

4 Single-Player Agents

Having a solid solution that can solve the single player levels is an important step during the cooperation levels since the agent should not be disturbed with planning and controlling difficulties that are not related to the cooperation itself. Both planning and control are presented as complex problems even for the single player levels due to the physics-based aspects of the game. Our agents first make a plan that can be either complete or partial. A plan contains a set of steps which indicate a position, velocity, the action that led to that position and velocity, and a number of diamonds, which correspond to a state the agent should arrive to. The controller then uses this information to guide the character, checking if the agent failed to reach a state at any time. When a partial plan is completed, or the agent fails to reach a state and recover from it, a new plan is made, repeating the process.

4.1 Planning

Geometry Friends is a game where planning has to be done in real time for it is not realistic to expect a human player will take more than a few seconds to start playing a level. Algorithm 1 shows the general idea of our version of RRT.

Algorithm 1. Adapted RRT

```
1: function PLAN_UPDATE
2:     if 𝒯 == null then
3:         BUILD_NEW_TREE();
4:     end if
5:     RUN_TREE(iterations);
6:     if goal != null OR (TIME_LIMIT() AND best_state != null) then
7:         return true;
8:     else
9:         return false;
10:    end if
11: end function
12: function BUILD_NEW_TREE
13:    𝒯.INIT(x_init);
14:    state_matrix.INIT();
15: end function
16: function RUN_TREE(iterations)
17:    for k = 1 to iterations do
18:        x_rand ← SELECT_STATE();
19:        while x_rand == null do
20:            EXPAND_STATE_MATRIX();
21:            x_rand ← SELECT_STATE();
22:        end while
23:        EXTEND(𝒯, x_rand);
24:    end for
25: end function
26: function EXTEND(x)
27:    u_new ← SELECT_ACTION();
28:    x_new ← APPLY_ACTION(x, u_new);
29:    if NEW_STATE(x_new) then
30:        𝒯.ADD_STATE(x_new);
31:        if BEST_STATE(x_new) then
32:            best_STATE ← x_new
33:        end if
34:        if GOAL_REACHED(x_new) then
35:            goal ← x_new
36:        end if
37:    end if
38: end function
```

Since there can be various levels of complexity, it is hard to choose a good number of iterations for the RRT algorithm that is not too low to be possible to find a solution, nor too high to waste time searching for a better solution when a good one has already been found. To avoid this, we run the search several times with a lower number, X, of iterations, and at the end of each search, it is checked if a solution was found, running another X iterations on the same tree if that is not the case. The value of X can be 1 or a value high enough to significantly

increase the changes of finding a better solution and be fast enough avoid having the agent staying still for more than a few seconds.

In an attempt to bias the search towards the goal we adapt the idea of LaValle et al. strategies of RRT-GoalBias and RRT-GoalZoom [7], during the algorithms state selection, along with Zickler et al. suggestion of the use of tactics and skills during the algorithms action selection [12]. For the tactics and skills, both agents have one tactic in a loop until all diamonds are caught, that can be divided in three stages: select diamond to collect; go to diamond platform; catch diamond. In a certain way, we divide the problem by assigning priorities when choosing a diamond to catch, since this will not compromise the solvability though, if not careful, might influence the solving time in a negative way. If there is a diamond in the same platform as the character, at a possible height for the character to catch, then this is the diamond to select. Otherwise the highest diamond on the highest platform is chosen since catching these first will not prevent to finish the level, in many cases.

We predict the application of an action on a state by using a simple forward model (predictor) for the circle and the game's own predictor for the rectangle. The predictors provided by the game are more accurate but too time expensive, hence the need to create our own. A simple model for the circle character was trivial to implement, though the same was not true for the rectangle hence keeping the original one. To avoid spending too much time searching around simple situations and too little on more complex ones, we use a dynamic time to predict the resulting state. The more complex a situation is, the shorter the time of the action simulation should be, so it can explore more states on the area around. We consider a complex situation, when an agent is on a short platform and rolling/sliding too much to the sides might make it fall when it should not. To choose how long the action simulation should be, we look at the platform width, and the shorter it is, less time is used for action simulation, meaning a bigger state exploration around the area. This strategy might be useful in any problem where it is possible to evaluate if a shorter time is needed, or if a longer time can be used. Another problem lies with the discretization of the state space. In our state space, a state is defined by the position of the agent, as well as its velocity, radius or height, depending if the agent is the circle or the rectangle, and a list of the uncaught collectibles. The state space of this problem is therefore theoretically infinite and so a discretization is needed for a fast search. To discretize we use a matrix of states, $Positions \times \#Diamonds$, to indicate whether or not a state has been reached. Though it is not the best representation, if the velocity and diamond ids are also considered, then the state space would still be big and the matrix consume too much memory. A state space that covers less of the real space can make the search faster but also impossible to solve. On the other hand, a big state space can slow down the search and disturb the level solving. Finding a balance that works for all kinds of levels is not trivial. To avoid this, we make the size of the search space dynamic by starting with a small matrix, dividing the width and height by a Y value, increasing it whenever the open list gets empty, and repopulating the

new matrix with the previous states, re-opening those which actions did not previously generate a new state so there is no need for a new search.

In problems like these, where an incomplete plan might make it impossible to reach the main goal, partial plans should be avoided but, in some situations, taking too long to find a solution might be worse, which is the case where there is a time limit to solve a level. Sometimes a choice must be made between risking to solve part of the problem or none at all. For this, we give our agents some time to search for a solution, and when that time is up, the best plan so far is followed. When that plan is completed, an attempt is made to find the rest of the solution.

4.2 Control

A controller, that can easily guide the agent to the correct positions of the plan at the desired velocities, is necessary for this problem. The RRT agents [9] use such controller which, in part, uses information the agents get when simulating their actions, and a Proportional Integral-Derivative (PID) controller which we tested for our solution which did not work as expected for it would sometimes fail to guide the character to the right side, or to obtain the required velocity to jump/morph or fall. A PID controller has several parameters that must be well tuned, and that might have been where we failed to make it work properly. To solve this we then used a new approach loosely based on the PID. This version uses the idea of acceleration to compute the direction the agent must take. The acceleration of the agent is calculated between the updates. When computing if the agent should go left or right, the controller checks first if the agent is already going in the right direction. If so, and if its velocity is higher than the desired one, it returns the action that goes to the opposite of the point side. If slower, it first simulated if the agent can actually reach the desired position with the desired velocity from the position it is currently in or if the it needs to go on the opposite direction first. When the circle agent reaches a waypoint, it checks if it needs to jump, doing so if the case. The rectangle morphing is more complex. When the next action is morph and the agent is not falling, it is verified if the agent has achieved the desired height. If not, the agent morphs up or down accordingly. In case the next point is right below the agent and not on the same platform, the agent, most likely is between two platforms and needs to morph up to fall through them. When the agent is falling, the current action in the plan is performed so it can morph up or down if needed.

4.3 Plan Correction and Replanning

The described controller can still fail to execute an action at the right position, making the characters land on a different platform than the one in the next step of the plan. Other times, the controller has difficulty to reach the desired velocity or position, and keeps failing the same action, over and over. One of the reasons for this might be due to our action simulation during the search, that is prone to errors, as previously mentioned. It is crucial for the agents to know when such

failed events happen, so they know they should stop following the current plan. To understand if a plan has failed, we give a time limit to reach the next step of the plan, as well as check if the agent is on the correct platform, meaning a platform from where it seems to be possible to reach the next step just by using the controller. If the time limit has been reached, or the agent is on a wrong platform then there is first an attempt to correct the plan. Sometimes, mainly when the agent falls into the wrong platform, that same platform might be on a step forward in the plan, which has the same number of caught diamonds. It also might happen to be in the same platform of a previous step, from which it might be useful to try again. If one of these steps is found, then the agent follows the plan from there. When no common step is found or the agent keeps failing to perform an action for n times, the agent then searches for a new plan.

5 Multiplayer Approach

The presented solution for single-player levels is enough to solve some cases where cooperative actions are needed. If there is a diamond where the other player needs only to position its character so the agent can use it as a platform, the agent is capable of finding a path if the other player positions the character at the right place. Even if the other player keeps its character still, the agent might find a way to push the other character to desired position to be possible to use as a platform, if possible. Though the problem does not involve the manipulation of objects, this solution is enough to be able to do so, in case these situations are added in.

Still, this is far from true cooperation and alterations are needed to create a version of the agents where they can solve more cooperative levels.

5.1 Planning

When searching for a solution, the agent now needs to simulate the actions of the partner besides their own, to know which changes the other character might do to the environment, as well as their possible positions. This means that, at each iteration of the RRT, the agent chooses a pair of actions, one of their own and one of their partner. This makes the state space of a multiplayer level exponentially bigger than of a single-player level, and the search time slower. The tactics and skills strategy can help reduce the number of times an action is uselessly selected and simulated and thus make the finding of a solution quicker. We keep the same tactic steps as in single-player mode: select diamond; go to diamond platform; catch diamond. While the main idea is the same, there is a difference in the last two steps. Going to a platform and catching a diamond has now skills that take into account cases where cooperation might be needed. Both agents, during the selection of an action, first call their skills to get their action, and with the resulting possible pair of actions which contain the selected one, call the skills of the other character to also get the action that might be more helpful in the current state. Currently, when following a plan, it is hard to understand

if an action is needed for cooperation or not, so removing seemly useless actions from the plan might make the it not viable. By this reason, when cooperating, plan cleaning is not performed bringing new challenges for the controller when executing a plan.

5.2 Replanning

In cooperative levels, not only the plans are more complex, the other player can also get in the way of the agent plan during the execution, which make the use of replanning a greater necessity. Simply starting a new tree can be significantly inefficient due to the bigger search space. Strategies like ERRT [3] and VLOD [13] reduce the time of replanning. The ERRT strategy uses bins of old states from the previous trees to select randomly during state selection for extension. We availed the idea of using bins but we use a different approach by keeping them to understand if one was reached after applying a pair of actions. If so, this means the corresponding node of the tree, along with its descendants, can be connected to the selected node in the new tree and time spent in simulation can be avoided. As suggested by the VLOD strategy, we should avoid replanning until it is actually necessary. In this solution, only the next few steps of the plan are used when verifying if the other player has turned into an obstacle.

6 Results and Discussion

6.1 Single Player and Agent-Agent Cooperative Players

We test the discussed variants of the RRT to understand which got better performance. Table 1, shows the results of the search time of the circle agent with different techniques, while using our forward model for the action simulation of the circle search and the game's own forward model for the rectangle. We selected three easy, Fig. 2, medium, Fig. 3, and hard levels, Fig. 4, ran each 10 times and present the time average (in seconds). The Original+STP variant indicates that the state selection is done as in the original RRT but the action selection uses tactics and skills. The combination with better performance, in average, was the BGT-Bias+STP which uses BK-BGT with a Bias to a goal state for state selection and tactis and skills for action selection. To test our complete single-player

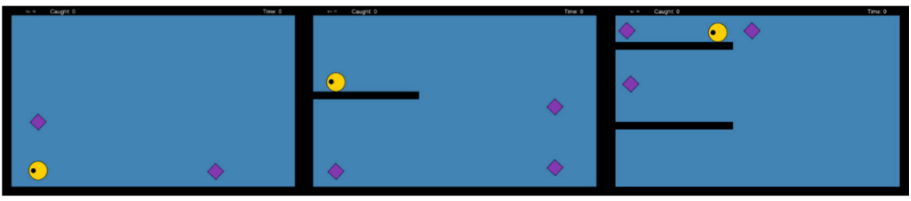

Fig. 2. Easy levels

Fig. 3. Medium levels

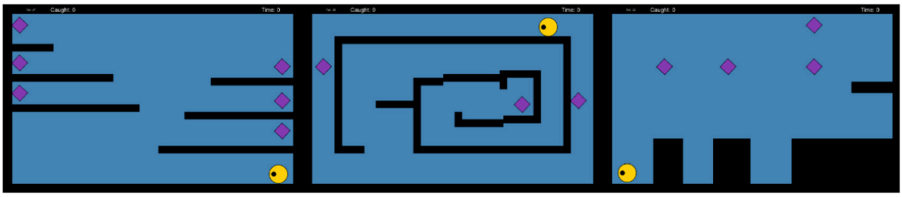

Fig. 4. Hard levels

and multiplayer solution, we compare our results to the contestants of the 2017 Geometry Friends Competition as well as some other available agents, as shown in Table 2.

Our agents rank in second place at the circle track and first at the rectangle and cooperative track, and though the latter is due to the lack of submitted agents for these tracks, it proves to overcome overspecialization. The cooperative track results are not good enough to conclude the agents can cooperate between themselves just with the solution as it is.

6.2 Human-Agent Cooperative Levels

To understand if our agents are currently capable of solving simple cooperative levels with human players, we prepared two sessions of tests. First we tested how human players worked as a team between themselves, to use as a base of comparison. We performed a set of tests with 8 two-player teams. The majority of the participants were casual players (61%), between 20 to 35 years old (92.3%), where almost half rarely or never play cooperative games (46.2%). Only two players had played Geometry Friends before this test. To simulate the lack of communication a human-agent team has, we asked our players to not communicate by

Table 1. Search time of the different RRT variants (average, in seconds)

Variant/Levels	Original	Bias	Zoom	BGT	BGT-Bias	Original+STP	BGT-Bias+STP
Easy	0.195	0.273	0.293	0.196	0.183	0.214	0.184
Medium	1.680	1.128	1.317	1.070	1.068	1.628	0.990
Hard	23.593	13.907	18.371	11.615	9.262	15.986	5.884

Table 2. Total scores of the agents on 2017 competition levels. C stands for Circle, R for Rectangle and Coop for Cooperative

Agent	Our Agents			KIT	RRT		Supervised	NRL	RL	Sub-Goal A*
Track	C	R	Coop	C	C	R	C	C	C	R
Points	3661	3139	2007	4203	1563	1234	1759	1925	2091	991

speaking or by using real world gestures, being possible to use the characters in an attempt to communicate. Then tests were executed with human-agent teams at a later stage. For this part, we first let the human players play with our agents without giving any information about them. After playing some levels, we asked their opinion about our agents through a questionnaire, and we then described their behavior, in a simple way, and asked to play again. This to understand if this new knowledge would change the opinion of the human players about our agents. Without communication, the human players showed to have difficulties to solve the levels as a team, as well as when they were playing with our agents. When asking for the players opinion about the agents level of intelligence, cooperation, and the players trust in the agent the answers were mainly positive, getting an average of 7 in a scale form 0 to 10 where we consider a negative response from 0 to 3, neutral from 4 to 6 and positive from 7 to 10. An interesting change happened after the players understood how the agents work. Though the players were more neutral regarding the rectangle (average of 6.5), many players changed their opinion about the circle agent for a more positive one (average of 7.8).

Even though our agents are not capable of differentiating a cooperative action from a simple waypoint in a plan, most human players believed the agents were cooperating with them. Having the agents rethink their plan, made the players believe the agents were adapting to them, even though they felt they add to adapt to the agents more times. Most players finished most levels under three minutes and, in most cases, the players felt their solving went well. We noticed that most players that had high expectations about the agents intelligence and capabilities do not work or study in any field of science.

7 Conclusions and Future Work

We address the problem of motion planning and execution in the Geometry Friends context which also challenges the agents to cooperate. Our proposed solution uses the algorithm RRT together with previously suggested strategies as well as new ones: use of dynamic discretization, dynamic prediction time for the action simulation, task priorities for biasing and partial planning. We also propose a controller that can follow said plan but can also understand when an action has failed and when it should try to recover using the same plan or re-plan from scratch. With this solution we are prepared to start to focus on the cooperation itself as future work.

Acknowledgements. This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT-UID/CEC/50021/2019) and through the project AMIGOS (PTDC/EEISII/7174/2014).

References

1. Bradshaw, J., Feltovich, P., Matthew, J.: Human-agent interaction. In: Boy, G. (ed.) Handbook of Human-Machine Interaction, pp. 283–302. Ashgate Publishing Ltd., Farnham (2011)
2. Bruce, J., Veloso, M.: Real-time randomized path planning for robot navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 3, pp. 2383–2388 (2002). https://doi.org/10.1109/IRDS.2002.1041624
3. Bruce, J., Veloso, M.: Real-time randomized motion planning for multiple domains. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 532–539. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74024-7_55
4. Hoffman, G., Breazeal, C.: Collaboration in human-robot teams. In: Proceedings of the AIAA 1st Intelligent Systems Technical Conference, pp. 1–18 (2004). https://doi.org/10.2514/6.2004-6434
5. Kraus, S.: Human-agent decision-making: combining theory and practice. Electron. Proc. Theoret. Comput. Sci. **215**, 13–27 (2016). https://doi.org/10.4204/EPTCS.215.2
6. LaValle, S.M.: Rapidly-exploring random trees: a new tool for path planning. Technical report, TR 98-11 (1998). https://doi.org/10.1.1.35.1853
7. LaValle, S.M., Kuffner, J.J.: Rapidly-exploring random trees: progress and prospects, pp. 293–308 (2000) https://doi.org/10.1017/CBO9781107415324.004
8. Prada, R., Lopes, P., Catarino, J., Quiterio, J., Melo, F.S.: The geometry friends game AI competition. In: Proceedings of 2015 IEEE Conference on Computational Intelligence and Games, CIG 2015, pp. 431–438 (2015)
9. Soares, R., Leal, F., Prada, R., Melo, F.: Rapidly-exploring random tree approach for geometry friends. In: Proceedings of 1st International Joint Conference of DiGRA and FDG (2016)
10. Van Wissen, A., Gal, Y., Kamphorst, B.A., Dignum, M.V.: Human-agent teamwork in dynamic environments. Comput. Hum. Behav. **28**(1), 23–33 (2012). https://doi.org/10.1016/j.chb.2011.08.006
11. Weibel, D., Wissmath, B., Habegger, S., Steiner, Y., Groner, R.: Playing online games against computer- vs. human-controlled opponents effects on presence, flow, and enjoyment. Comput. Hum. Behav. **24**(5), 2274–2291 (2008). https://doi.org/10.1016/j.chb.2007.11.002
12. Zickler, S., Veloso, M.: Efficient physics-based planning: sampling search via non-deterministic tactics and skills. In: The 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 27–34 (2009)
13. Zickler, S., Veloso, M.: Variable level-of-detail motion planning in environments with poorly predictable bodies. Frontiers Artif. Intell. Appl. **215**, 189–194 (2010). https://doi.org/10.3233/978-1-60750-606-5-189