Visualisation of a physical model of interacting of hard objects in a computer game

E V Soboleva\(^1\) and E V Kharunzheva
Department of Digital Technologies in Education, Vyatka State University, Kirov, Russia

\(^1\)E-mail: sobolevaelv@yandex.ru

Abstract. The problem of this research consists of relevant need to develop a physical model of interaction of hard objects, and herewith physical modelling of interaction of hard objects is often used in computer games. The purpose of this research is visualization of a physical model of interaction of hard objects in a computer game. Methodology of this research consists of analysing of domestic and foreign authors' scientific researches: method of system-information analyse, method of mathematical and informational modelling: method of computer test. As a result of research the existing methods realizing collisions of objects in computer games are analyzed; the model of a game engine is developed. The algorithms in this model were visualised and tested. In conclusion, there are inferences, which confirm that dynamics of an absolute hard and deformable agents, modelling of gas and liquid are basic part of any physical process.

1. Introduction

Today, the value system of many people is focused on nature consumption and subordination to itself. Calculation power of computers grows every day to give an opportunity to develop different physical principles. Due to this modelling of physical phenomena has a rising popularity in films and computer games. The big part of users compares phenomena of physic in real life with events in films and computer games. In this regard, realistic modelling physical phenomena is actual branch in visual practical-Converter activity [1].

There is a special software – a physical engine – for creating physic phenomena in computer programs. The physical engine is a program, creating model of real physic phenomena [2]. The physics engine is independent module and is a part of game engine or is used in imitation modelling of physical phenomena. We need to find a compromise between the accuracy of the calculations and execution speed when to create a physics engine according to the task.

Creation of a physical engine is hard task, even if it has narrow functionality.

Despite a lot of physical phenomena simulated by physical engines (dynamics of absolutely hard and deformable agents, modeling of liquids and gases, action of tissues, etc.) – dynamics of absolutely hard agent is the basic component of any physical engine. In process of modeling phenomena in the studied cases, the main task is to detect and resolve collisions of interacting objects.

Goals and objectives of the research. Thus, the purpose of the research is determined from the need to solve the problem of visualisation of the physical model of interaction of hard agents. As an application area of the model, a computer game is chosen as one of the most promising areas in technical and scientific innovation [3].
To achieve the goal, the following tasks were set:

- make analysis of existing methods of implementing collision in computer games;
- develop an informational model of the game physics engine;
- visualization of interaction models and implement algorithms that correspond to the selected methods of the description of collisions;
- test and debug the developed program.

2. Materials and Methods

Theoretical methods – analysis of psycho-pedagogical, scientific and technical literature, software development for the implementation of collision detection, algorithms implementing the interaction of agents in computer games.

The method of system-information analysis was used to analyze algorithms for detecting narrow phase collisions and to select collision resolution methods and methods for integrating equations of motion. The method of mathematical and informational modeling, the method of computer experiment, as a kind of computational used to describe the game application, the development of a hard agent model, at the stages of modeling hard station physics and visualization of the model of the physical engine.

Collision detection is the computational problem of analyzing or detecting the intersection of objects \[4\]. Basically, the problem of detection is often solved in the process of development of computer games \[5\]. However, the research area affects many other scientific fields. The indicated computational difficulty is used in automated design, for the synthesis of trajectories of the cutting tool in numerical control systems, for programming the movement of robots in an environment with obstacles, in virtual prototyping systems, in computer modeling of physical processes, etc. \[6\]. In computer-aided design of assembly processes, this problem is called geometric access or geometric solvability \[4\].

There are two main ways to defining collisions: a priori and a posteriori. In a posteriori way, the scene analysis for collision detection is performed at short intervals of time. In a priori approach, it is necessary to calculate the trajectory of objects and predict before the collision (a priori) with static elements of the scene, taking into account the friction forces, the collision elasticity, as well as changes in the internal state of the deformable objects \[7\]. In General, this problem is described by a system of differential equations and may not have an exact analytical solution, and its numerical solution requires significant computational resources. Thus, in the a priori approach, the fact of collision detection is determined before the origin of the collision.

In this research, a posteriori approach to the detection of columns is chosen, as it is widely used in practice, and there is a possibility of detecting collisions of objects in real time, which there is no possibility to make it in a priori approach. The main problem in usage of a posteriori way is resource-intensive algorithms that handle the collision \[8\]. It is obvious that the computational complexity increases with the number of processed objects. In this regard, there is a problem of optimization of applications which use such algorithms.

One of the solutions to this problem is in Browne C and Maire F \[7\]. Its essence is the separation of modeling interactions of objects in three phases: broad phase, narrow phase, and resolution of conflict \[9\].

Modeling of hard agent physics is divided into several parts: detection of intersection of parallelepipeds, limiting objects; detection of intersection of polygons of objects; collision resolution; calculation of forces acting on the object; integration of equations of motion.

According to the results of the literature analysis, the following algorithms were identified for modeling the interaction of agents:

- The brute force algorithm, the Spatial Hashing algorithm, and the Sweep-and-Prune algorithm were studied for the broad phase stage and for detecting the intersection of AABB objects. In this research, the choice was made in favor of the Sweep-and-Prune algorithm, as algorithm has
a better time estimate of $O(n \log n)$ due to the sorting of objects along the axis with the highest
density of objects than the Brute force algorithm based on the search of all pairs of objects.

- For the part of the narrow phase and detection of the intersection of polygonal grids of objects,
algorithms were studied: an algorithm based on the separating axis theorem, the Gilbert-Johnson
Keerthi algorithm + Expanding Polytope Algorithm, The Algorithm of Lin-Canny, Algorithm
V-Clip. After considering narrow phase detection algorithms, the choice was made in favor of
an algorithm based on the separating axes theorem.

- For the part of collision resolution it is possible to use projection algorithms, calculation of
momentum, calculation of elastic forces. The narrow phase determines the collision, when the
objects have already intersected, but on the scene interpenetration of objects should not occur.
Therefore, a correction of the position of the objects that the prima value method of projection,
and then applies the method of calculation of pulses.

- For the integration part of the equations of motion, we generalize the algorithm: Explicit Euler
Integrator, Implicit Euler Integrator, Improved Euler Integrator, 4th order Runge-Kutta
Integrator, Time Adjusted Verlet Integrator. However, the improved Euler method is stable and
most optimal.

3. Results
The game application will be aimed at cross-platform use. Java was chosen as the programming
language thanks to the existing cross-platform library libgdx. Android Studio was chosen as the
development environment [10].

Description of the game application. To test the model of the physical game engine, a game
application in the genre of 2d platformer was developed [3]. Platformer – a genre of computer games in
which the main feature of the gameplay is jumping on platforms, climbing stairs, picking up items that
are usually necessary to complete the level.

Description of the solid model. Objects in the physics engine can be represented by a convex polygon
or a circle. For simplicity, we will use the term geometry in cases where it does not matter what shape
the object has. An object represented as a convex polygon is defined by a set of vertices, where each
vertex has its own weight and is defined by coordinates on the plane. An object represented by a circle
is defined by the radius and center of the circle [1].

All objects are considered to be absolutely solid. A solid agent is an agent that does not change its
shape. Each object has physical properties: density $p$, mass $m$, rotation angle, speed $u$, acceleration $a$,
angular speed, angular acceleration, inertia of the agent, torque, a set of forces $F$, coefficient of elasticity
$e$, coefficient of friction $k$. Each object is affected by the force of attraction. In a collision, objects
exchange elastic impulses, taking into account rotation and friction [2].

Description of the stages of the physics simulation of a rigid agent. At the first stage, a wide phase is
realized. The selected Sweep-and-Prune method allows to significantly optimize the collision detection
algorithms of solids. Sweep-and-Prune is a method of sorting bounding parallelepipeds by coordinates.
Using this algorithm requires a bounding box (AABB) for all objects. Thus, it is required to make
limiting parallelepipeds for all objects.

Detection of intersection between the AABB objects are realized by means of algorithm of Sweep-
and-Prune. The algorithm at the input receives a list of all objects of the game world. Next, you want to
calculate the sample variance for each axis.

The algorithm at the input receives a list of all objects in the game world. Next, you need to calculate
the sample variance for each axis. The variance will be calculated by the formula:

$$D[X] = M[X^2] - (M[X])^2$$

where $X$ – is the arithmetic mean of the values $\min (x, y)$ and $\max (x, y)$,
$M[X]$ – expected value.
At the second stage, a narrow phase is realized. From the analysis it was found that the most effective algorithm for creating a 2d physics engine, based on Separating Axis Theorem, SAT using the optimization proposed by Dirk Gregorius in 2013, which is to find the reference points [11].

Implementation of the method: the initial position and speed are stored, the acceleration from the position and speed is calculated. Then the position \( p_2 \) and speed \( v_2 \) are calculated. Next, we calculate the acceleration from the position and speed.

\[
\begin{align*}
  p_1 &= position; \\
  v_1 &= velocity; \\
  a_1 &= acceleration(p_1, v_1); \\
  p_2 &= p_1 + v_1 \times time; \\
  v_2 &= v_1 + a_1 \times time; \\
  a_2 &= acceleration(p_2, v_2); \\
  position &= (v_1 + v_2) \times time / 2; \\
  velocity &= (a_1 + a_2) \times time / 2;
\end{align*}
\]

The third step is the calculation of all forces acting on the object based on the detected contact points. For this stage of the research is used the method of calculation of impulses.

At the fourth stage, there comes a change in the positions of objects relative to the calculated acting forces and impulses. From the analysis it was found that the stable and most optimal is Improved Euler Integrator is a second order integrator, i.e. the accumulated error has the order of the second derivative.

Thus, the possibility of developing a module describing the geometric shape of the object, as well as a module that determines the physical properties of the object. The need to develop the following modules for collision detection is derived: two convex polygons, two circles, a convex polygon, and a circle. After detecting collisions, you need to develop a module to exchange pulses between a pair of objects. It is obvious that the visualized physical model requires a controller that controls all the stages of modeling.

When developing a physical engine, you should think carefully about the overall structure of the application. During the development of the physical engine model, the need for the development of the following classes was identified:

- **Shape.** An abstract object shape class that combines the Circle and Polygon classes. Contains common fields and methods for inheriting classes.
- **Circle.** This class is created as a separate independent module that defines the geometry of the object. Inherited from the Shape class. Used to represent a circle. At the entrance gets the radius and center of the circle.
- **Polygon.** This class is created as a separate independent module that gives geometry to the object. Inherited from the Shape class. Used to represent the convex polygon of an object. At the input receives an array of points that form a convex polygon.
- **PhysicsObject.** The class is created as a separate independent physical module that defines the physics of the object.
- **GameActor.** A game object that contains the geometry of the Shape class and the physics of the PhysicsObject class.
- **Contact.** Contains links to two traversed the object, the point of con-tact, the vector and the minimum depth of penetration.
- **SATCollisionCallback.** The interface contains two overloads of the intersection check method.
- **SATPolygonToPolygon.** The class implements the interface methods SATCollisionCallback, for two convex polygons on the basis of SAT.
- **Satscicletocircle.** The class implements the SATCollisionCallback interface methods for two sat-based circles.
SATCircleToPolygon. The class implements the interface methods SATCollisionCallback, for the circle and a convex polygon on the basis of SAT.

SATPolygonToCircle. In the class override the invocation order of the objects for the class SATCircleToPolygon.

ContactSolver. The class takes data from the Contact class and resolves collisions by applying pulses to contact points, calculating linear and angular velocity, friction force.

PhysicsController. The class performs the role of updating physics. The object pairs are checked for intersection using the algorithm for the wide phase. Then, for all potentially colliding depending on the geometric shape of the object determines their intersection using the appropriate algorithms of the narrow phase. If objects intersect, the narrow phase algorithms return a list of contacts between pairs of objects. Next, the ContactSolver class methods are run to calculate the pulses at the contact points, the PhysicsObject class methods change the speeds and positions of objects in accordance with the calculated pulses and the acting forces on the object.

We present the results of each phase of the collision of two objects. For one of the tests the collision of two objects of square shape was realized. One object fell from above under the influence of gravity on another object standing on the platform. The parameter of elasticity was determined equal to one (absolutely elastic collision). After detecting the intersection of objects in the wide phase, the objects were checked for intersections in the narrow phase by the Separating Axis Theorem algorithm. The last stage of the collision was the resolution of the collision. After a collision between objects, elastic pulses are exchanged. The linear and angular velocity is calculated. Pulses applied to the contact points cause the object standing motionless on the platform to move.

The procedure of multiple and complete testing of the developed physics engine for modeling the interaction of solids showed full efficiency.

4. Conclusion
The aim of the study was to visualize the physical model of interaction of solids on the example of a computer game. To achieve this goal, the analysis of collision detection and resolution algorithms, as well as methods for integrating the equations of motion. As a programming tool we used object-oriented language Java, as an environment Android Studio. The cross-platform library libgdx was used to visualize the application. The paper describes a model of a game object that includes solid state physics and geometric representation: a circle or a convex polygon. Also, the module of interaction of solids, which is divided into several stages: wide, narrow phase, collision resolution, integration of the equation of motion.

During the design, the need to create the following classes: a class that determines the geometric shape of the object, a class of solid state physics, a class of game object, a class that controls all stages of physics, collision detection classes for a convex polygon and a circle, a class containing information about the detected collision, a class for the transmission of pulses between a pair of colliding agents.

The formulated algorithm for collision detection and contact point detection was fully visualized. During the full multi-fold testing, it is proved that the points of contact between a pair of colliding objects are determined correctly and are used for the exchange of elastic pulses. A number of optimizations that significantly reduce the computational load of the application was presented.

Thus, the efficiency of the developed physical engine is confirmed and the effectiveness of the proposed optimizations is proved.

Acknowledgments
This research is funded by Russian Science Foundation under Grant No 18-78-10053.

References
[1] Aslamova E A, Krivov M V and Aslamova V S 2018 Vestn. Tom. state un-that. Manag. Comp. Sci. 44 84
[2] Haşıloğlu M A, Keleş P U and Aydin S 2011 Proced. - Social Behav. Sci. 28 1053
[3] Ioannou A and Makridou E 2018 *Edu. Inf. Technol.* **23** (6) 2531
[4] Kazantsev G V 2018 *News Tomsk Polytech. Univ. Geores. Eng.* **329** (8) 112
[5] Soboleva E V, Karavaev N L, Shalaginova N V and Perevozchikova M S 2018 *Euro. J. Contemp. Edu.* **7** (4) 845
[6] Makryniotis T 2014 Game design with the aid of computer semiotics *Journal of Gaming and Virtual Worlds* **6**(3) 255–73
[7] Browne C and Maire F 2010 Evolutionary game design *IEEE Transactions on Computational Intelligence and AI in Games* **2**(1) 1–16
[8] Soboleva E V, Galimova E G, Maydangalieva Z A, and Batchayeva K K 2018 Didactic value of gamification tools for teaching modeling as a method of learning and cognitive activity at school *Eurasia Journal of Mathematics, Science and Technology Education* **14**(6) 2427–44
[9] Umarov I and Mozgovoy M 2014 Creating believable and effective AI agents for games and simulations: Reviews and case study *Contemporary Advancements in Information Technology Development in Dynamic Environments* 25
[10] Pirovano M and Lanzi P L 2014 Fuzzy tactics: A scripting game that leverages fuzzy logic as an engaging game mechanic *Expert Systems with Applications* **41**(13) 6029–38
[11] Hernández-Orallo J, Martínez-Plumed F, Schmid U, Siebers M and Dowe D L 2016 Computer models solving intelligence test problems: Progress and implications *Artificial Intelligence* **230** 74–107