Qualitative Physics in Angry Birds: first results

Przemysław Wałęga\(^1\) and Tomasz Lechowski\(^2\) and Michał Zawidzki\(^3\)

Abstract. Angry Birds is a well known game in which the main aim is to destroy all pigs located in a given level using a slingshot and birds of various kinds. In most cases pigs are positioned among other objects - building blocks, rocks or specifically profiled ground - whose behaviour reflects the laws of physics. The game rewards chariness of a player - the less birds she uses and the more damage she does to the whole structure in the level, the higher score she gets.

In our paper, we attempt to indicate the essential factors playing role in the process of destruction. Thus, we provide definitions of influence (which we further divide into two categories - horizontal impact and vertical impact), stability and connection points. Whereas the latter two can be evaluated regardless of what shot we will actually take, the former requires an assumption about the shot and it is counted by means of iterative analysis involving two central concepts - propagation of force (in the case of horizontal impact) and center of rotation (in the case of vertical impact).

The ultimate goal of the whole procedure is assigning each object which is reachable by a shot a numerical value expressing the scale of damage it does once hit. The playing program is then supposed to shoot at the object with the highest value.

1 Introduction

Angry Birds game gained popularity thanks to its simple rules and the behaviour of the game environment reflecting the laws of physics. The main aim in each level is, having a bundle of birds (sometimes of various kinds) at hand, to kill all the pigs, doing as much damage to other objects as possible. Usually pigs are not exposed to direct shots - in these situations one first has to get through sheltering structures protecting a pig. Since the number of birds we have at our disposal (our ammo) is each time limited, we need to carefully select points we want to shoot at. A bad choice can result in a pig surviving the whole cannonade. If we take a closer look at difficulties the game can cause to players of both types - human and AI - it turns out that what is the toughest task for a human agent - hitting the selected target - for an AI agent is a matter of routine. Indeed, having the object picked, a program can easily attain angle and power of the shot using patterns known from simple Newtonian physics. The choice of an object worth targeting at is, however, much tougher for an AI agent, notwithstanding the fact that humans often make it intuitively.

Several most notable examples are: \(\text{RCC} - 8\) ([7]), Allen’s Interval Algebra (primarily introduced as representation of the flow of time, see [1]), Rectangle Algebra ([2]) or the 4-Intersection Method ([3]). Despite their qualitative nature (which could presumably imply that they resemble human reasoning) they struggle with certain common difficulties, namely relatively low expressive power. In the context of such a complex problem like modelling behaviour of a set of objects in a dynamic environment they are insufficient in most cases. Thus, both \(\text{RCC} - 8\) and the 4-Intersection Method can only express topological information (A overlaps B, A is disconnected from B etc.), whereas Rectangle Algebra - captures both topological and directional properties of rectangles (like, e.g., to be above or to be to the left, cf. [8, 9]) but leaves “untouched” the issue of mutual physical impact of the objects (for instance, A has a horizontal impact on B of value n) which usually can be evaluated in a dynamic context. A rather fruitful attempt of exploiting an extended version of the Interval Algebra and Rectangle Algebra can be found in [8, 9]. However, these tools are the most efficient in the case of blocks positioned orthogonally to two main axes. Usually such layout can only be found in an initial situation (before the first shot) in each level. After the first shot, often introducing a high level of entropy to the construction of blocks, most objects are positioned slantwise.

In this paper we focus not only on the initial layouts of block constructions but we also take into account more messy arrangements after the shots being taken. The main goal we want to achieve is attributing a unique numerical value to each object reachable by a shot. Consequently, object with the highest number - representing its highest strategic value - is chosen to be shot at in the first place. Our approach is both qualitative and quantitative. Qualitative because we qualitatively pick factors which, in our opinion, play the key role in the process of destruction carried out in the game. Hence, we distinguish and scrutinize the “static” notions: stability (involving the notions of mass and relative location) shelter, being above or touching and the “dynamic” notions: horizontal and vertical impact (with, respectively, force propagation and center of rotation as their central concepts). The process of selection of such key values is of qualitative nature as well as assigning them weights when calculating their overall impact on the whole arrangement. Quantitative factor of the approach is expressed by a numerical character of all patterns used to count the final value of a block. Certain inspirations from naive or qualitative physics (cf. [4, 6, 5]) can also be traced in our proposal.

2 Representation

Since the final choice of the best shot in an Angry Birds gameplay is based mainly on objects’ spatial configuration, it follows that space representation of Angry Birds level plays a significant role here. The vision module provided by Angry Birds Competition organizers enables to extract all objects from a gameplay scene, i.e. to determine
their contours, center points, etc. Our approach extends this quanti-
tative representation and involves more abstract qualitative relations
such as: “object \textit{o}\textsubscript{1} touches \textit{o}\textsubscript{2}”, “object \textit{o}\textsubscript{1} is above \textit{o}\textsubscript{2}” or “object
\textit{o}\textsubscript{1} is a shelter for \textit{o}\textsubscript{2}”. Such qualitative representation is more natu-
ral for human cognition and seems to be essential for a human while
playing Angry Birds.

2.1 Quantitative representation

The provided vision module enables us to calculate the following
quantitative data:

- \( O = \{ o\textsubscript{0}, o\textsubscript{1}, \ldots \} \) – the set of all objects, i.e. the set containing
  blocks, pigs, hills and ground, where each object \( o\textsubscript{i} \) is identified
  with points inside \( o\textsubscript{i} \), i.e. \( o\textsubscript{i} = \{ p\textsubscript{i} | p\textsubscript{i} \in o\textsubscript{i} \} \).
- \( \text{center}(o\textsubscript{i}) \) – the center point of \( o\textsubscript{i} \) object, i.e. the center of \( o\textsubscript{i} \) mass
- \( \text{area}(o\textsubscript{i}) \) – an area of \( o\textsubscript{i} \) object,
- \( x(p) \) – the \( x \) coordinate of a point \( p \),
- \( y(p) \) – the \( y \) coordinate of a point \( p \),
- \( \text{dist}(p\textsubscript{1}, p\textsubscript{2}) \) – distance between points, i.e. the number of pixels
  between \( p\textsubscript{1} \) and \( p\textsubscript{2} \),
- \( \text{dist}(o\textsubscript{1}, o\textsubscript{2}) \) – distance between an object \( o\textsubscript{1} \) and an object \( o\textsubscript{2} \), i.e.
  \( \text{dist}(o\textsubscript{1}, o\textsubscript{2}) = \min_{p\textsubscript{1} \in o\textsubscript{1}, p\textsubscript{2} \in o\textsubscript{2}} \text{dist}(p\textsubscript{1}, p\textsubscript{2}) \),
- \( \text{dist}(p, o\textsubscript{1}) \) – distance between a point \( p \) and an object \( o\textsubscript{1} \), i.e.
  \( \text{dist}(p, o\textsubscript{1}) = \min_{p\textsubscript{1} \in o\textsubscript{1}} \text{dist}(p, p\textsubscript{1}) \),
- \( \text{width}(o\textsubscript{i}) \) – width of a Minimal Bounding Rectangle (MBR) for
  an object \( o\textsubscript{i} \) – see Fig. 1,
- \( \text{height}(o\textsubscript{i}) \) – height of a Minimal Bounding Rectangle (MBR) for
  an object \( o\textsubscript{i} \) – see Fig. 1,
- \( \text{object\textunderscore type}(o\textsubscript{i}) \) – a type of an object \( o\textsubscript{i} \), i.e. \textit{ice}, \textit{wood}, \textit{stone} or
  \textit{pig},
- \( B = \{ b\textsubscript{1}, b\textsubscript{2}, \ldots \} \) – the set of all birds,
- \( \text{bird\textunderscore type}(b) \) – a type of a bird \( b \), i.e. \textit{red}, \textit{blue}, \textit{yellow}, \textit{white}
or \textit{black}.

\( o\textsubscript{i} \) is a shelter for \( o\textsubscript{0} \), i.e. \( o\textsubscript{0} \) reserved for ground
\( O\textsubscript{1}, O\textsubscript{2}, \ldots \) for sets of objects,
\( p, p\textsubscript{0}, p\textsubscript{1}, p\textsubscript{2}, \ldots \) for points.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{object}
\caption{The object \( o\textsubscript{i} \), its center of mass and MBR’s height and width.}
\end{figure}

Additionally, the trajectory module provided by the organizers en-
ables us to determine \( \text{trajectory}(o\textsubscript{i}) = \{ \text{trajectory}(p\textsubscript{0}), \text{trajectory}(p\textsubscript{1}) \} \), i.e.
the set of 2 trajectories (low and high) of a shot targeted at the
center of \( o\textsubscript{i} \), where \( \text{trajectory}(o\textsubscript{i}) = \{ p\textsubscript{0}, p\textsubscript{1}, p\textsubscript{2}, \ldots \} \) is a set of
points that belong to the low parabola trajectory and \( \text{trajectory}(o\textsubscript{i}) = \{ p\textsubscript{0}, p\textsubscript{1}, p\textsubscript{2}, \ldots \} \) is a set of points that belong to the high parabola
trajectory.

In the paper, we use the following notation:

- \( o\textsubscript{0}, o\textsubscript{1}, o\textsubscript{2}, \ldots \) for objects, with \( o\textsubscript{0} \) reserved for ground
- \( O\textsubscript{1}, O\textsubscript{2}, \ldots \) for sets of objects,
- \( p, p\textsubscript{0}, p\textsubscript{1}, p\textsubscript{2}, \ldots \) for points.

- \( B \) for the set of all birds,
- \( b, b\textsubscript{0}, b\textsubscript{1}, b\textsubscript{2}, \ldots \) for birds.
- \( T \) for the set of all trajectories,
- \( t, t\textsubscript{0}, t\textsubscript{1}, t\textsubscript{2}, \ldots \) for trajectories.

2.2 Qualitative representation

We define \textit{touch} – a binary relation between objects. Intuitively, two
objects \textit{touch} each other whenever they have a common point. How-
ever, since the computer vision component of the game playing soft-
ware does not always precisely identify objects’ location, we also
consider objects that do not have any common point, but the distance
between them is smaller than a constant \( c \) (\( c \) is about 2 pixels), being
in \textit{touch} relation – see Definition 3.

\begin{definition}[\textit{touch} relation]
\[ \forall o\textsubscript{1}, o\textsubscript{2} \in O (o\textsubscript{1} \neq o\textsubscript{2} \rightarrow (\text{touch}(o\textsubscript{1}, o\textsubscript{2}) \equiv \text{dist}(o\textsubscript{1}, o\textsubscript{2}) < c)) \]
\end{definition}

Obviously, the \textit{touch} relation is irreflexive and symmetric. As an
example consider the block structure presented in Fig. 2, where \( o\textsubscript{1}
\) touches \( o\textsubscript{2} \), \( o\textsubscript{2} \) and \( o\textsubscript{0} \) (ground), therefore the following statements
are true: \( \text{touch}(o\textsubscript{1}, o\textsubscript{2}), \text{touch}(o\textsubscript{1}, o\textsubscript{4}), \text{touch}(o\textsubscript{1}, o\textsubscript{3}) \).

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{block}
\caption{A sample block structure.}
\end{figure}

For any two objects \( o\textsubscript{1}, o\textsubscript{2} \) that are in \textit{touch} relation, we deter-
mine a set of connection points i.e. a set of points that are simulta-
neously very close to \( o\textsubscript{1} \) and \( o\textsubscript{2} \) (closer than the constant distance \( c \)).
More precisely, we define a function \( \text{connection\textunderscore p} : O \times O \rightarrow\{ p\textsubscript{0}, \ldots, p\textsubscript{n} \} \), as presented in Definition 2.

\begin{definition}[\textit{connection\textunderscore p} function]
\[ \forall o\textsubscript{1}, o\textsubscript{2} \in O (\text{connection\textunderscore p}(o\textsubscript{1}, o\textsubscript{2}) = \{ p | \text{dist}(p, o\textsubscript{1}) < c \land \text{dist}(p, o\textsubscript{2}) < c \}) \]
\end{definition}

\begin{definition}[\textit{touch} relation]
\[ \forall o\textsubscript{1}, o\textsubscript{2} \in O (\text{touch}(o\textsubscript{1}, o\textsubscript{2}) \equiv (o\textsubscript{1} \neq o\textsubscript{2} \land \text{connection\textunderscore p}(o\textsubscript{1}, o\textsubscript{2}) \neq \emptyset)) \]
\end{definition}

Furthermore, we determine 3 significant points that belong to
\( \text{connection\textunderscore p}(o\textsubscript{1}, o\textsubscript{2}) \), i.e. the leftmost, rightmost and center con-
nection points, by means of functions presented in Definitions 4, 5
and 6.
Definition 4 (left_p function).

\[
\begin{align*}
\forall x(p) &= \min_{p_i \mid p_i \in \text{connection}_p(o_1, o_2)} x(p_i) \\
\forall y(p) &= \frac{1}{2} \left( \min_{\forall x(p_i) = x(p)} y(p_i) \right) \\
&\quad + \max_{\forall x(p_i) = x(p)} y(p_i)
\end{align*}
\]

Definition 5 (right_p function).

\[
\begin{align*}
\forall x(p) &= \max_{p_i \mid p_i \in \text{connection}_p(o_1, o_2)} x(p_i) \\
\forall y(p) &= \frac{1}{2} \left( \min_{\forall x(p_i) = x(p)} y(p_i) \right) \\
&\quad + \max_{\forall x(p_i) = x(p)} y(p_i)
\end{align*}
\]

Definition 6 (center_p function).

\[
\begin{align*}
\forall x(p) &= \frac{1}{2} \cdot (x(\text{left_p}(o_1, o_2)) + x(\text{right_p}(o_1, o_2))) \\
\forall y(p) &= \frac{1}{2} \left( \min_{\forall x(p_i) = x(p)} y(p_i) \right) \\
&\quad + \max_{\forall x(p_i) = x(p)} y(p_i)
\end{align*}
\]

As an example consider the structure presented in Fig. 3, where \(o_1\) touches \(o_2\), therefore we can determine the set of connection points \(\text{connection}_p(o_1, o_2)\) and among them: the leftmost, rightmost and center connection points. In this example we have \(p_1 = \text{left_p}(o_1, o_2), p_2 = \text{right_p}(o_1, o_2)\) and \(p_3 = \text{center_p}(o_1, o_2)\). Note also that in the case horizontally adjacent object \(\text{left_p}\) and \(\text{right_p}\) collapse to a single point, namely the point located in the “middle” of the \(\text{connection}_p\) set. It is justified by the fact that we only distinguish between particular connection points when we are searching for rotation centres, which is inherently connected with vertically adjacent objects.

\[
\begin{array}{c}
\bullet \quad p_1 \quad \bullet \\
\bullet \quad o_1 \quad \bullet \\
\bullet \quad o_2 \quad \bullet \\
\bullet \quad p_3 \quad \bullet \\
\bullet \quad p_2 \quad \bullet
\end{array}
\]

Figure 3: left_p, center_p and right_p connection points between objects \(o_1\) and \(o_2\).

We define the upper edge of an object \(o_1\) as a set of points \(\{p_0, \ldots, p_n\}\) such that for each \(i \in \{0, \ldots, n\}\), \(p_i\) belongs to the \(o_1\) contour and a point directly above \(p_i\) does not belong to the \(o_1\). The upper edge of an object is determined by the function \(\text{upper_edge} : O \rightarrow \{p_0, \ldots, p_n\}\) defined below.

Definition 7 (upper_edge function).

\[
\forall o_1 \in O(\text{upper_edge}(o_1) = \{p \in o_1 | \forall p_1(p) = (x(p), y(p) + 1) \rightarrow p_1 \notin o_1\}).
\]

A pictorial presentation of the upper_edge of objects of various shapes is presented in Fig. 4.

\[
\begin{array}{c}
\begin{array}{c}
\text{Figure 4: upper_edge for objects with various shapes.}
\end{array}
\end{array}
\]

Now we can define above – a binary relation between objects. The intuitive meaning of above\(\langle o_1, o_2 \rangle\) is that \(o_1\) lies on \(o_2\), i.e. \(o_2\) is pressed by \(o_1\) weight. A formal definition of the above relation is given below.

Definition 8 (above relation).

\[
\forall o_1, o_2 \in O(\text{above}(o_1, o_2) \equiv \text{touches}(o_1, o_2) \land \text{center_p}(o_1, o_2) \in \text{upper_edge}(o_2))
\]

The above relation is obviously irreflexive and asymmetric. It should be clarified that \(\text{above}(o_1, o_2)\) is not equivalent to the statement that \(o_2\) is a support for \(o_1\), e.g. in Fig. 2 we have \(\text{above}(o_1, o_2)\) but at the same time \(o_1\) is a support of \(o_2\) (even though, due to asymmetry of above, it is not the case that \(o_2\) is above \(o_1\). The above relation will play a significant role in our reasoning algorithm, therefore it needs to be understood correctly and cannot be confused with a support relation. We will denote a transitive closure of the above relation by \(\text{above}^*\), which is obviously irreflexive, asymmetric and transitive.

2.3 Shelters

In Angry Birds it often appears that there is no other way of hitting a pig than destroying its sheltering structure first. It is then reasonable to define sheltering objects and to distinguish the most important shelter. We define shelter – a ternary relation between two objects and a trajectory which denotes the fact that the first object is a shelter for the second object (always a pig) with respect to a given trajectory. Intuitively, \(\text{shelter}(o_1, o_2, t)\) means that we cannot directly shoot a pig \(o_2\) using trajectory \(t\), because \(o_1\) stands in the way leading to \(o_2\), i.e. \(o_1\) is situated on the trajectory \(t\) before \(o_2\). More precisely, \(\text{shelter}(o_1, o_2, t)\) whenever \(o_2\) is a pig and \(o_1\) lies on one of trajectories estimated by the trajectory module for a shot aiming at \(o_2\). A formal definition of the shelter relation is as follows.

Definition 9 (shelter function).

\[
\forall o \in O \exists t \in T(t \in \text{traj}(o) \land \text{pig}(o)) \rightarrow \text{shelter}(o, t) = [o_1, o_2, o_n]
\]

where \(o_n\) is located nearest to \(o\) and the lower the index of an element in the sequence, the further the element is located from \(o\).

Furthermore, for each object we determine a shelter value which allows us to distinguish between more and less important shelters.
We define a function $shelter\_val: O \rightarrow \mathbb{R}$ that maps an object into its shelter value. A shelter value of $o_1$ increases as the distance to the sheltered pig $o_2$ decreases, the number of objects between $o_1$ and $o_2$ decreases or the number of pigs sheltered by $o_1$ increases. The formal definition of $shelter\_val$ is presented below.

**Definition 10 (shelter\_val function).**

\[
\forall o_1 \in O \left(shelter\_val(o_1) = \sum_{o_2 \in \text{shield}(o_2,t)} \max_{o_2 \in \text{no\_between}(o_1,o_2,t)} \frac{1}{\text{dist}(o_1,o_2) \text{no\_between}(o_1,o_2,t)+1}\right),
\]

where $\text{no\_between}(o_1, o_2, t)$ is a number of objects between $o_1$ and $o_2$ on trajectory $t$.

As an example consider Fig. 5. On a trajectory $t_1$, there are 2 shelters for the $o_4$ pig, namely $o_3$ and $o_2$. However, $o_3$ has the highest shelter\_val because it is closest to the $o_4$ pig and there is no other shelter on the trajectory $t_1$ between $o_3$ and $o_2$.

![Figure 5: $o_2$ and $o_3$ are shelters for the $o_4$ pig, while considering $t_1$ shooting trajectory.](image)

### 2.4 Stability

Stability is a quality of an object, which represents how hard it is to move it, e.g., by hitting it with a bird or another object. The object's stability increases whenever the number and stability of objects directly below $o_1$ increases, $o_1$ ratio of width to height increases, the cumulated mass of $o_1$ and all objects above $o_1$ increases, and a cumulated mass of the objects directly to the right of $o_1$ increases. Formal definition of a function $\text{stability}: O \rightarrow [0,1]$ that assigns each object a stability value from the interval $[0,1]$ (where 0 means that the object will move without any stimuli, while 1 means that the object is impossible to move) is presented below.

**Definition 11 (stability function).**

\[
\forall o_1 \in O \left(\text{stability}(o_1) = \left(\frac{\text{card}(\{o_2 \mid \text{above}(o_1,o_2)\})}{\sum_{o_2 \in \text{above}(o_1,o_2)} \text{stability}(o_2)} \cdot \text{ratio}(o_1) \cdot \text{mass}(o_1) \right)^\frac{1}{\max_{o_2 \in \text{above}(o_1,o_2)} \text{area}(o_2)} \right),
\]

where

\[
\forall o_1 \in O \left(\text{ratio}(o_1) = \begin{cases} \frac{\text{width}(o_1)}{\text{height}(o_1)} & \text{if } \text{width}(o_1) \geq \text{height}(o_1) \\ \text{otherwise} \end{cases} \right),
\]

\[
\forall o_1 \in O \left(\text{mass}(o_1) = \frac{1}{\sum_{o_2 \in \text{above}(o_1,o_2)} \text{area}(o_2)} \right),
\]

\[
\forall o_1 \in O \left(\forall o_2 \in \text{on\_right}(o_1,o_2) \equiv \text{touch}(o_1,o_2) \wedge x(\text{connection}_p(o_1,o_2)) > x(\text{center}(o_1)), \right)
\]

and $p(o_1): O \rightarrow [0,1]$ maps an object into qualitative classes of density, namely $\rho_{\text{ice}}, \rho_{\text{wood}}, \rho_{\text{stone}}, \rho_{\text{pig}}$.

### 3 Reasoning

In order to choose the best shot possible, we reason as follows. We have decided that there are two main types of objects which are of much interest to us. First and foremost we are interested in pigs, but we are also interested in blocks which constitute shelters for the pigs. We assign a value to each object based on how it influences above-mentioned objects (see section 3.3 for more details). We choose the object which has the highest value as our target. We calculate the influence of one object on another by means of its vertical impact and horizontal impact which will be discussed in the following two sections.

#### 3.1 Vertical Impact

In many cases blocks are placed in such a way, that hitting one of them leads to a collapse of a number of others. In order to determine such an influence one need to consider how blocks are located with respect to each other. While quantitative method of calculating exact forces values that appears in a structure requires a lot of effort and afterwards leads to a high computational complexity reasoning algorithms, we have established our own qualitative physics method. Our method, called Vertical Impact, considers what would have happen if one of blocks in a structure would disappear (fall or be destroyed). Consider a situation presented on Fig. 6 a), i.e. a situation when the $o_1$ block is hit, and as a result it is destroyed or falls down. The question is if other blocks remain stable or also fall down?

![Figure 6: Vertical Impact: hitting $o_1$ causes $o_3$, $o_4$, $o_7$, $o_6$ falling down.](image)
are in relation above star with $o_2$) falls iff $p_1$ is to the left of $p_1$ or to the right of $p_2$. In our example it is the case that $x(p_3) < x(p_1)$, so objects from the set $O_1$ will fall down, i.e. they are added to the list fall and marked on Fig. 6 (b) with dashed crosses. In fact, it is not the end of the algorithm, because, there are still other objects that may fall down too. For every object from $O_2$ (without $o_1$) we recursively launch Algorithm 2 which checks how the cumulated center of mass of a given block and all blocks that are in relation above star with it but are not in fall list is located. If it is to the left of the leftmost or to the right of the rightmost connection point then the object falls. Afterwards, we check recursively other objects that are under already considered block. In our case we need to check $o_2$ and $o_6$ – see Fig. 6 b). While $o_2$ remains table, $o_6$ will fall, since $x(p_3) > x(p_2)$. It is still not the end of the algorithm, because, we need to check all objects that are below $o_2$ and $o_6$ (marked with a blue spline), in this case just $o_7$ which obviously will remain stable. Finally, we launch recursively Algorithm 2 for objects that are under $o_1$ but in our case there are no such objects. The output of the algorithm in a given example is a list of objects that fall, namely $fall = \{o_1, o_3, o_4, o_5, o_6\}$.

Algorithm 1 Vertical impact

Input: object $o_1$
initializelist $fall$ with one element i.e. $o_1$
for all $o_2 \mid above(o_2, o_1)$ do
if $x(center(o_2) \cup \bigcup o_3 \mid above*(o_1, o_2)) \leq min_{o_3\mid above*(o_2, o_3), o_3 \neq o_1} x(left_p(o_2, o_1))$ or $x(center(o_2) \cup \bigcup o_3)$ above star $(o_3, o_2) \geq max_{o_3\mid above*(o_2, o_3), o_3 \neq o_1} x(left_p(o_2, o_3))$ then
add $o_2$ to $fall$
for all $o_3 \mid above*(o_3, o_2)$ do
add $o_3$ to $fall$
for all $o_4 \mid \exists o_3((above*(o_3, o_2) \land above(o_3, o_4) \land o_4 \notin fall) \lor above(o_1, o_4))$ do$fall_2 = output from Algorithm 2 with inputs: o_4, fallfall = fall \cup fall_2$Output: list $fall$ of objects falling if $o_1$ will fall

Algorithm 2 Recursive falling checking

Input: object $o_1$, list of falling objects $fall$ if $x(center(o_1) \cup \bigcup o_2\mid above*(o_2, o_1) \land o_2 \notin fall)$ \leq min_{o_3\mid above*(o_1, o_3), o_3 \neq o_1} x(left_p(o_1, o_3))$ or $x(center(o_1) \cup \bigcup o_2\mid above*(o_2, o_1) \land o_2 \notin fall)$ \geq max_{o_3\mid above*(o_1, o_3), o_3 \neq o_1} x(right_p(o_1, o_3))$ then
add $o_1$ to $fall$
for all $o_2 \mid above(o_1, o_2) \land o_2 \notin fall$ do$fall_2 = output from Algorithm 2 with inputs: o_2, fallfall = fall \cup fall_2$Output: updated list of falling objects $fall$

3.2 Horizontal Impact

The second method – called Horizontal Impact – enables to reason about force propagation between objects. Notice, that hitting an object $o_1$ with a bird, affects also on objects touching $o_1$. We will (naively) say that the force propagates to nearby blocks. As an example consider a situation presented on Fig. 7, where $o_3$ is directly hit by a bird with a force $F_o$. Horizontal Impact method will enable us to estimate (qualitatively) $F_o$ force and determine how it propagates to nearby objects. At first, the force will be propagated to direct neighbours, i.e. $o_1$, $o_4$ and $o_5$. Afterwards the force will be propagated also to $o_2$ and $o_6$ and then also to $o_7$ but the propagation is not infinite and there will be no significant force affecting on $o_8$. Since the propagating force always decreases, we have established a minimal force $F_{min}$. If the estimated force is lower then the minimal level, force propagation is stopped – for details see Definition 13.

![Figure 7: A Horizontal Impact method for determining force propagation after direct hitting of $o_3$.](image)

The first step in the Horizontal Impact method is to estimate the force (a value from interval $[0,1]$) affecting on a directly hit object – $o_3$ in our example. For simplicity, we have decided that the direct (horizontal) force depends only on the choice of trajectory - it is higher for the low trajectory. - see Definition 12.

Definition 12 (Direct force).

∀$o_1 \in O, t \in T(F_{o_1}(o_1, t) = traj_type(t, o_1))$ and $traj_type : T \times O \rightarrow [0, 1]$ is a following function:

∀$t \in T, \forall o_1 \in O(traj_type(t, o_1) = \begin{cases} 1 & \text{if } t = traj_{-1}(o_1) \\ 0.7 & \text{if } t = traj_{-1}(o_1) \end{cases})$

Afterwards, the force is propagated to objects that are in touch relation with a previous object. Consider object $o_1$ in touch relation with $o_2$ and force $F_{o_1}$ affecting $o_1$ that propagates to $F_{o_2}$ affecting $o_2$. The value of $F_{o_2}$ force increases whenever $F_{o_1}$ increases or the stability of $o_1$ decreases. Formally, propagated force is described in Definition 13.

Definition 13 (Propagated force).

$F_{o_2} = loc_{coefficient}(1 - stability(o_1))^2$, where the $loc_{coefficient}$ distinguishes between the objects which are directly to the right of $o_1$, that are directly above and those directly below. The propagated force is the highest in the first case ($loc_{coefficient} = 1$), smaller in the second case ($loc_{coefficient} = 0.8$) and even smaller in the last case ($loc_{coefficient} = 0.2$). Notice that each force has a value from an interval $[0,1]$. After determining the force propagation we know what forces affect objects and therefore, we can investigate their effects. Namely, we can deduce, if a given force destroy an object $o_1$, knock it down or the object $o_1$ remains unmovd. The effect of the force is deduced just on the
basis of the affected object mass, i.e. if \( F_{o_1} \geq f_d(\text{area}(o_1) \cdot \rho(o_1)) \) – the force is larger than a value of \( f_1 \) function of \( o_1 \) mass, then \( o_1 \) will be destroyed, otherwise if \( F_{o_1} \geq f_f(\text{area}(o_1) \cdot \rho(o_1)) \) then \( o_1 \) will be fall down, and otherwise \( o_1 \) remains unmoved, where \( \forall a \in \mathbb{R}[f_d(a), f_f(a) \in [0, 1] \land f_d(a) > f_f(a)] \). Notice, that if \( F_{o_1} \geq f_f(\text{area}(o_1) \cdot \rho(o_1)) \) for some object \( o_1 \), then \( o_1 \) is destroyed or falls down. Hence, \( o_1 \) may have a vertical influence on other blocks and Vertical Impact algorithm presented in Section 3.1, namely Algorithm 1 with \( o_1 \) as an input needs to be lunched. As an example consider once more a situation from Fig. 7. Let’s assume that \( o_2 \) is hardly hit with a bird and as a result, it influences significantly \( o_3 \) with a force \( F_{o_3} \) but \( o_3 \) itself remain unmoved. Let us assume now that \( F_{o_2} \geq f_f(\text{area}(o_2) \cdot \rho(o_2)) \) and as a result \( o_3 \) falls down. Therefore, we need to check if \( o_3 \) has a vertical impact on other blocks. As presented in Section 3.1, we need to check if the center of \( o_3 \) mass is between it’s leftmost and rightmost connection points. In our case it is not, therefore we can conclude that \( o_3 \) will fall also. The algorithm does not stop in this moment but works as presented earlier in previous section in the Algorithm 2. The described algorithm for Horizontal Impact is presented in Algorithms 3 and 4.

**Algorithm 3** Horizontal impact

**Input:** directly hit object \( a_0 \), fired bird \( b \), trajectory \( t \) of a shot initialize an empty list \( \text{fall} \) of falling objects initialize a list \( \text{forces} \) of forces affecting objects, where \( \text{forces}[k] \) indicates a force affecting \( o_k \), i.e. \( F_{o_k} \). Initially for each \( k \), \( \text{forces}[k] = 0 \)

if \( \text{forces}[i] \geq f_f(\text{area}(o_i) \cdot \rho(o_i)) \) then add \( o_i \) to \( \text{fall} \)

for all \( o_j \in \text{fall} \) do

\( \text{forces} = \text{forces} \cup \text{forces} \)

for all \( o_j \in \text{fall} \) do

\( \text{forces} = \text{forces} \)

Output: list \( \text{fall} \) of falling objects, list \( \text{forces} \) of forces affecting objects

**Algorithm 4** Recursive force propagation

**Input:** objects \( a_0, a_j \), list \( \text{fall} \) of falling objects, list \( \text{forces} \) of forces affecting objects

if \( \text{forces}[i] \geq F_{\text{min}} \) then

\( \text{forces}[i] = \text{forces}[i] \cdot \text{local} \_ \text{coeff} \cdot (1 - \text{stability}(o_i))^2 \)

add \( o_j \) to \( \text{fall} \)

for all \( o_k \in \text{touch}(o_i, o_j) \) do

\( \text{forces} = \text{forces} \)

\( \text{forces} = \text{forces} \)

Output: list \( \text{fall} \) of falling objects, updated list \( \text{forces} \) of forces affecting objects

Afterwards, we determine an influence value of a directly hit object to another object according to a given bird and shot trajectory. More precisely, we introduce a function \( \text{influence} : O \times O \times B \times T \rightarrow [0, 1] \) presented in Definition 14.

**Definition 14** (influence function),

\[ \forall o_1, o_2 \in O, b \in B, t \in T(\text{influence}(o_1, o_2, b, t)) = \begin{cases} 1 & \text{if } o_1 \in \text{fall} \text{ otherwise.} \end{cases} \]

where \( o_1 \) is a directly hit object, i.e. an input to the Algorithm 3.

Obviously, \( \text{influence}(o_1, o_2, b, t) = 0 \) means that while shooting at \( o_1 \) with a bird \( b \) and trajectory \( t \), \( o_1 \) has no influence on \( o_2 \), whereas \( \text{influence}(o_1, o_2, b, t) = 1 \) means that if \( o_1 \) will fall, then undoubtedly \( o_2 \) will fall also.

### 3.3 Value estimation

In order to choose the best shot in a given Angry Birds gameplay we need to estimate an overall value of available shots. This value depends on the type of the bird that will be lunched, type of the shooting trajectory (high or low parabola) and a block which is a direct target of the shot. More precisely, the value of a shot, i.e., the value of a function \( \text{value} : B \times T \times O \rightarrow \mathbb{R} \) increases whenever the stability of a direct target of a shot decreases or the influence of a directly hit object on interesting objects increases, where interesting objects are pigs and their shelters. For the exact description see in Definition 15.

**Definition 15** (value function),

\[ \text{value}(b, t, o_1) = \text{corr} \_ \text{fact}(o_1, b, t) \cdot \sum_{o_2 \in O} \text{ivalue}(o_2) \cdot \text{influence}(o_1, o_2, b, t) + \text{shape} \_ \text{coeff} \cdot \text{shape} \_ \text{value}, \]

where \( \text{ivalue} : O \rightarrow \mathbb{R} \) maps an object into its interesting value as follows:

\[ \forall o_1 \in O(\text{ivalue}(o_1)) = \begin{cases} 1 & \text{if } o_1 \text{ is a pig} \text{ otherwise.} \end{cases} \]

and the \( \text{corr} \_ \text{fact} \) is takes different values depending on the combination of the type of the bird and the type of material of the given object. This factor takes into account the ability of certain birds to destroy certain blocks (for instance yellow bird is very prolific efficient against wooden blocks). For the exact values of \( \text{corr} \_ \text{fact} \) see the table below:

<table>
<thead>
<tr>
<th>bird_type</th>
<th>object_type</th>
<th>pig</th>
<th>ice</th>
<th>wood</th>
<th>stone</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>1</td>
<td>0.7</td>
<td>0.8</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>blue</td>
<td>1</td>
<td>1</td>
<td>0.1</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>yellow</td>
<td>1</td>
<td>0.1</td>
<td>1</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>white</td>
<td>1</td>
<td>0.7</td>
<td>0.3</td>
<td>0.6</td>
<td></td>
</tr>
<tr>
<td>black</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Based on the fact that spheres play a significant role on certain game levels, we have decided to increase their value in the following way. If a sphere is located higher than some pig and there is a straight
path (without any obstacles) leading from the sphere to that pig we increase the value of the sphere. We further increase the sphere’s value for each additional pig on the given path. We only count one such path (the one with the most pigs on it’s way). The \textit{shape value} returns the highest number of pigs on the path leading from the given sphere to any pig and \textit{shape coeff} (\(= 0.4\)) scales down this number, so that it is still pays off better to shoot directly at a pig than to aim at a sphere that would kill that pig. The provided trajectory module enables to determine directly reachable objects. After calculating reachable objects, we indicate a best shot as a one with a highest value among reachable shots. Notice, that since we consider only shooting at center points of objects, the number of available and reachable shots is small enough to perform all presented calculations in a reasonable time.

4 Evaluation

We have tested our algorithm on the first twenty one levels of the Poached Eggs scenario. If the algorithm failed to complete a given level, we repeated that level, but only once. If it failed to complete a given level twice, we would move on to the next level. It took our algorithm 33 minutes to go through all the 21 levels. It failed to complete 2 levels. The total score recorded was 753080, which gives an average score of 35861 per level. We also run the Naïve Naive Agent under the same conditions. It took the Naïve Agent 34 minutes to go through all the levels. It failed to complete 3 levels. The total score recorded was 687490, which gives an average score of 32738 per level.

5 Conclusions and future work

We have outlined a procedure for reasoning about the choice of a target in an Angry Birds game. The procedure involves several notions, which are central to the reasoning, such as stability, shelter, impact, influence. We choose our target to be the object that has a low stability, but it influences (by having a high, either vertical or horizontal, impact) objects which are pigs or shelters of pigs. Further testing is required to fine-tune the coefficients used in formulas for calculating numerical values of these notions (for instance the density of building materials). The procedure shows promise, but it has its flaws. There are certain configurations of objects, which serve as counter-examples to our reasoning. However these configurations are mostly theoretical possibilities as they seldom occur in the actual game. Still there is a lot of work to be done in order to expand our reasoning as to include the abilities of certain birds (for instance the ability to drop an egg of the white bird) and to include the behaviour of blocks of certain shapes (spheres or certain complex polygons). There is also potential for extending the procedure to specify how and where a given block will fall if it falls. In other words one can use the above-mentioned concepts to reason not only that a given block will fall but that it will fall, say, to the right and hit another block. All of this can be done using the introduced notion of connection points.

ACKNOWLEDGEMENTS

The work presented in this paper is supported by the Polish National Science Centre grant 2011/02/A/HS1/00395.

REFERENCES