SoftwareX article template Version 4 (November 2023)

Before you complete this template, a few important points to note:

- This template is for an original SoftwareX article. If you are submitting an update to a software article that has already been published, please use the Software Update Template found on the Guide for Authors page.
- The format of a software article is very different to a traditional research article. To help you write yours, we have created this template. We will only consider software articles submitted using this template.
- It is mandatory to publicly share the code/software referred to in your software article. You'll find information on our software sharing criteria in the SoftwareX Guide for Authors.
- It's important to consult the Guide for Authors when preparing your manuscript; it highlights mandatory requirements and is packed with useful advice.

Still got questions? Email our editorial team at softwarex@elsevier.com.
Now you are ready to fill in the template below. As you complete each section, please carefully read the associated instructions. All sections are mandatory, unless marked optional. Once you have completed the template, delete these instructions. In addition, please delete the instructions in the template (the text written in italics).

Title (Name of your software: Then a short title)

A. Author1[a], A. Author2[b]

[a] *Author1's affiliation, address, email*
[b] *Author2's affiliation, address, email*

Abstract

Ca. 100 words. The abstract is followed by a maximum of six keywords and some mandatory and optional metadata.
Your main body of text (sections 1-5 below) should be a maximum 6 pages in total (excluding metadata, tables, figures, references) with a 3000-word limit (we ask that more priority is placed on the word limit versus the page count). Though we strictly insist on the author following the template, in exceptional circumstances, we can be flexible with the page numbers and word limit. In such cases, it should be discussed with the managing editor or publisher prior to submission. All queries regarding the same can be reached at softwarex@elsevier.com.

Keywords: keyword 1, keyword 2, keyword 3

Metadata

The ancillary data table 1 is required for the sub-version of the codebase. Please replace the italicized text in the right column with the correct information about your current code and leave the left column untouched.
Optionally, you can provide information about the current executable software version filling in the left column of Table 2. Please leave the first column as it is.

1. Motivation and significance

In this section, we want you to introduce the scientific background and the motivation for developing the software.

- *Explain why the software is important and describe the exact (scientific) problem(s) it solves.*
- *Indicate in what way the software has contributed (or will contribute in the future) to the process of scientific discovery; if available, please cite a research paper using the software.*

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v1.0
C2	Permanent link to code/repository used for this code version	`https://github.com/marborawski/DSSTowerDefense` `https://github.com/marborawski/TowerDefense`
C3	Permanent link to Reproducible Capsule	For example: `https://codeocean.com/capsule/0270963/tree/v1`
C4	Legal Code License	GNU General Public License GPL-3.0
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	C#, Matlab (Octave), Unity
C7	Compilation requirements, operating environments & dependencies	
C8	If available Link to developer documentation/manual	`https://github.com/marborawski/DSSTowerDefense/tree/main/Doc/`
C9	Support email for questions	mariusz.borawski@usz.edu.pl

Tabela 1: Code metadata (mandatory)

- *Provide a description of the experimental setting. (How does the user use the software?)*
- *Introduce related work in literature (cite or list algorithms used, other software etc.).*

2. Software description

Describe the software. Provide enough detail to help the reader understand its impact.

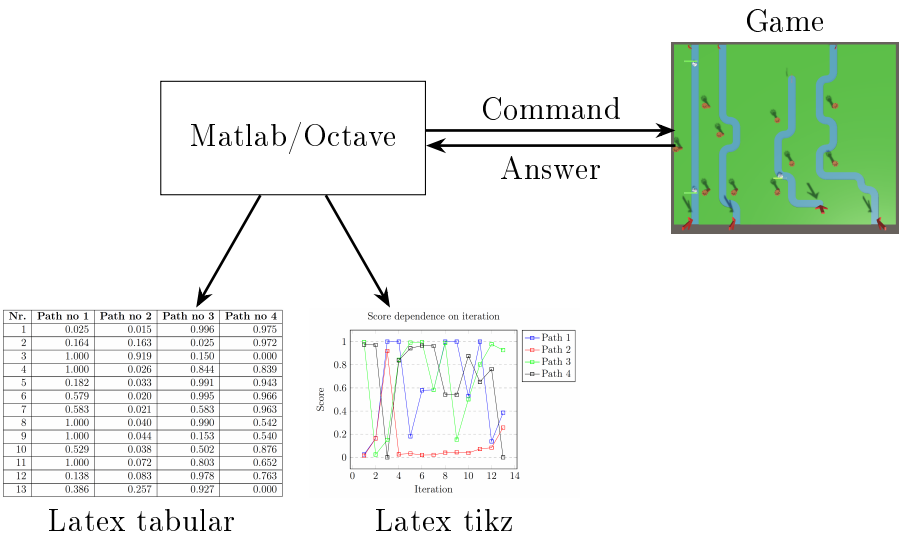
2.1. Software architecture

Oprogramowanie GTest wykorzystuje platformę programistyczną Matlab/Octave oraz silnik gier Unity. Część przeznaczona dla silnika gier jest napisana w języku C#. Implementuje mechanikę rozgrywki w grę typu tower defense. Rolę gracza pełni oprogramowanie napisane w języku Matlab. Komunikacja między oprogramowaniem w Matlab-ie a grą odbywa się na zasadzie kilent-serwer (fig. 1). Gra oczekuje na polecenia nasłuchując na wybranym porcie. Możliwa jest komunikacja zdalna poprzez internet. Oprogramowanie w Matlab-ie

Nr.	(Executable) software metadata description	Please fill in this column
S1	Current software version	For example 1.1, 2.4 etc.
S2	Permanent link to executables of this version	For example: `https://github.com/combogenomics/DuctApe/releases/tag/DuctApe-0.16.4`
S3	Permanent link to Reproducible Capsule	
S4	Legal Software License	List one of the approved licenses
S5	Computing platforms/Operating Systems	For example Android, BSD, iOS, Linux, OS X, Microsoft Windows, Unix-like , IBM z/OS, distributed/web based etc.
S6	Installation requirements & dependencies	
S7	If available, link to user manual - if formally published include a reference to the publication in the reference list	For example: `http://mozart.github.io/documentation/`
S8	Support email for questions	

Tabela 2: Software metadata (optional)

jest zbiorem funkcji pozwalających na realizację badań metod decyzyjnych. Umożliwia pobieranie od gry aktualnych informacje o jej stanie i pozwala na przekazywanie jej komend sterujących realizujących akcje stawianie nowych wież na planszy oraz wypuszczania nowych przeciwników.
Rysunek 2 przedstawia ważniejsze klasy gry tower defense. Jedną z najważniejszych jej klas jest UnityServer realizująca wymianę informacji w formacie XML z oprogramowaniem działającym na platformie programistycznej Matlab. Instancja tej klasy dekoduje otrzymane rozkazy i w zależności od potrzeb komunikuje się z innymi obiektami przekazując im zadania i zbierając potrzebne dane. Klasa ManagementTilemap zajmuje się zarządzaniem najmniejszymi elementami mapy – tile. Instancja klasy ManagementTilemap zleca instancji klasy CreateTilemap wykonanie tilemap w oparciu o dostępne tile. Gromadzi informacje statystyczne związane z wydarzeniami na kafelkach takich jak przejście przeciwnika, zabicie przeciwnika itp. Przechowuje informacje o różnych typach tile, a w tym o ich wyglądzie (ma dostęp do ich modeli 3D). Zleca instancji klasy ManagementCost wyznaczenie optymalnej trasy ruchu minimalizującej koszt przejścia od punktu startowego do



Rysunek 1: Architektura systemu

końcowego. Instancja klasy Towers przechowuje informacje o wieży, jej parametrach oraz wyglądzie. Podobnie instancja klasy EnemyInfo przechowuje informacje o przeciwniku, jego parametrach oraz wyglądzie. Instancja klasy ManagementCash zajmuje się zarządzaniem cash. Wyznacza ile cash mają wieże, ile przeciwnicy i jest odpowiedzialna za prezentowanie tej informacji na HUD-ie i jej aktualizację.

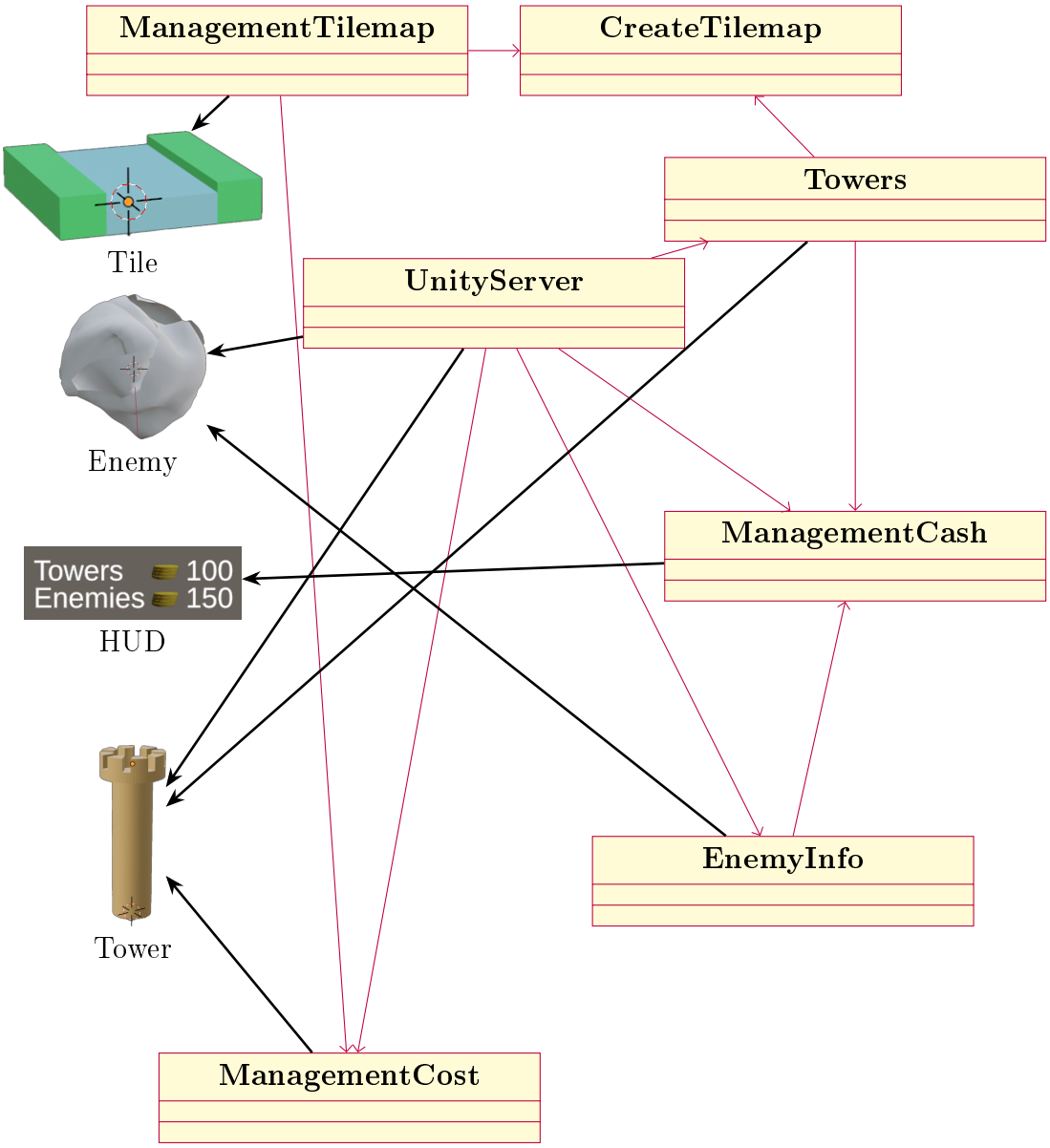
Give a short overview of the overall software architecture; provide a pictorial overview where possible; for example, an image showing the components. If necessary, provide implementation details.

2.2. Software functionalities

Funkcjonalności oprogramowania GTest :

1. Tworzenie planszy gry typu tower defense z dowolną liczba tras przejścia;
2. Możliwość planowania czasu i miejsca dostawiania wież;
3. Możliwość decydowanie przez metody decyzyjne którą ścieżką będą poruszać się przeciwnicy;
4. Tworzenie statystyk związanych z przejściami przeciwników przez ścieżkę;
5. Tworzenie tabel i wykresów umożliwiających porównanie badanych metod.

Projektowanie planszy gry wymaga stworzenia modeli przeciwnika, wieży oraz kafelków (terenu i drogi). Modele takie można stworzyć wykorzystując



Rysunek 2: Software architecture: uproszczony diagram klas z dodanymi obiektami 3D powiązanymi z klasami

dowolne oprogramowanie do modelowania obiektów 3D. Przykładem takiego oprogramowania jest Blender zaliczany do kategorii wolnego i otwartego oprogramowania. Podstawowy zestaw modeli został już wykonany i dołączony do projektu Unity. Są to modele do gry tower defense w której wieże strzelają do śmieci płynących rzeką. Jeżeli istnieje taka konieczność modele te można zmienić z poziomu edytora Unity.
Plansza gry jest tworzona na podstawie przesłanych jej danych w formie XML-a. Zawierają one informacje o rodzaju kafelków (teren, droga) i punktach startowych oraz końcowych. Wizualizacja planszy jest tworzona automatycznie z dostępnych modeli 3D.
Postawienie wieży jest realizowane w momencie otrzymania przez grę komendy zapisanej w formacie XML-a. Do komendy dołączana jest informacja o położeniu wieży. Podobnie wysłanie przeciwnika jest realizowane poprzez wysłanie komendy z dołączonymi informacjami o punktach startowych i końcowych ścieżki po których będzie się on poruszał.
W każdej chwili można grze wysłać polecenie przesłanie informacji o stanie gry. Informacje zwracane są formacie XML-a, dotyczą między innymi długości ścieżek, liczby wież przy ścieżkach, liczby przeciwników, którzy wystartowali z punktu startowego, liczby przeciwników, którzy dotarli do punktu końcowego, poziom życia z jakim dotarli do punktu końcowego, liczby pocisków, które trafiły przeciwnika na określonej ścieżce, liczby pocisków, które nie trafiły przeciwnika na określonej ścieżce, dostępny cash dla przeciwników, dostępny cash dla wież. Możliwe jest także wysłanie komendy z informacjami uszczegółowionymi, które zawierają dodatkowo położenie wszystkich kafelków dróg, parametry wież i parametry przeciwników.
Biblioteka dla środowiska Matlab/Octave zawiera głównie funkcje ułatwiające komunikację grą. Przygotowuje dane w formacie xml dla tworzenia planszy, stawiania nowych wież i wypuszczania przeciwników. Pozwala na dekodowanie danych w formie XML do postaci ułatwiającej manipulowanie nimi w Matlab-ie. Dodatkowo biblioteka zawiera metody pozwalające na konwersję tablicy do środowiska tabular Latex-a oraz formy akceptowalnej przez moduł tworzenia wykresów pakietu tikz-a.
Present the major functionalities of the software.

2.3. Sample code snippets analysis (optional)

3. Illustrative examples

Provide at least one illustrative example to demonstrate the major functions of your software/code.
***Optional**: you may include one explanatory video or screencast that will appear next to your article, in the right hand side panel. Please upload any video*

as a single supplementary file with your article. Only one MP4 formatted, with 150MB maximum size, video is possible per article. Recommended video dimensions are 640 x 480 at a maximum of 30 frames / second. Prior to submission please test and validate your .mp4 file at `http://elsevier-apps.sciverse.com/GadgetVideoPodcastPlayerWeb/verification` *. This tool will display your video exactly in the same way as it will appear on ScienceDirect.*

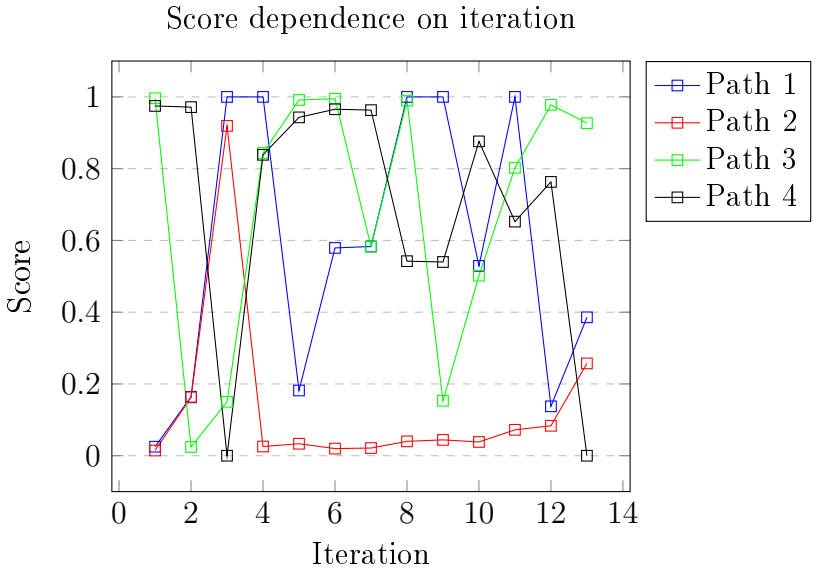
Nr.	Path no 1	Path no 2	Path no 3	Path no 4
1	0.025	0.015	0.996	0.975
2	0.164	0.163	0.025	0.972
3	1.000	0.919	0.150	0.000
4	1.000	0.026	0.844	0.839
5	0.182	0.033	0.991	0.943
6	0.579	0.020	0.995	0.966
7	0.583	0.021	0.583	0.963
8	1.000	0.040	0.990	0.542
9	1.000	0.044	0.153	0.540
10	0.529	0.038	0.502	0.876
11	1.000	0.072	0.803	0.652
12	0.138	0.083	0.978	0.763
13	0.386	0.257	0.927	0.000

Tabela 3: Test

4. Impact

This is the main section of the article and reviewers will weight it appropriately. Please indicate:

- *Any new research questions that can be pursued as a result of your software.*
- *In what way, and to what extent, your software improves the pursuit of existing research questions.*
- *Any ways in which your software has changed the daily practice of its users.*
- *How widespread the use of the software is within and outside the intended user group (downloads, number of users if your software is a service, citable publications, etc.).*



Rysunek 3: Test

- *How the software is being used in commercial settings and/or how it has led to the creation of spin-off companies.*

Please note that points 1 and 2 are best demonstrated by references to citable publications.

5. Conclusions

Acknowledgements

Optional. You can use this section to acknowledge colleagues who don't qualify as a co-author but helped you in some way.

Literatura

[1] Use this style of ordering. References in-text should also use a similar style.

If the software repository you used supplied a DOI or another Persistent IDentifier (PID), please add a reference for your software here. For more guidance on software citation, please see our guide for authors or this article on the essentials of software citation by FORCE 11, of which Elsevier is a member.

Reminder: Before you submit, please delete all the instructions in this document, including this paragraph. Thank you!