

A Sample Project Template for Spiltek Students

Elev En, Elev To, Elev Tre, Elev Fire, Elev Fem

Abstract

This document is a template for Spiltek project reports. Your abstract should briefly state the game or system you built, the approach you took, and your main result — for example, what a playtest revealed about your core mechanic.

Index Terms

game design, playtesting, prototyping, level design

I. INTRODUCTION

LaTeX is a typesetting system in which you write plain text marked up with simple commands, and it produces a polished, consistently formatted document. Its main advantage is that you concentrate on *what* you are writing while LaTeX handles *how* it looks: it manages spacing, fonts, section numbering, and the placement of figures (see Fig. 1) and tables for you, so the layout takes care of itself and stays uniform throughout. The rest of this document introduces the pieces you will need for a Spiltek report.

This is a normal paragraph. In LaTeX you do not press Enter to start a new paragraph. Instead, a paragraph continues until you leave a completely blank line; a single line break in the source is treated as an ordinary space, so you can wrap your source text however you like without affecting the output. One or more blank lines end the current paragraph and begin the next one (extra blank lines have no additional effect). Avoid using `\\` to separate paragraphs: that command forces a line break *within* a paragraph and does not start a new one, so it usually produces the wrong spacing.

This is the second paragraph. Notice the blank line above it in the source and the first-line indentation LaTeX adds automatically to mark the paragraph break.

LaTeX allows basic formatting:

Bold text

Italic text

Emphasised text

Some characters have a special meaning in LaTeX and cannot simply be typed as-is. The ten reserved characters are `# $ % & _ { } ~ ^ \`. To print seven of them literally, put a backslash in front: `\# \$ \% \& \_ \{ \}`. The remaining three are trickier: use `\textasciitilde` for `~`, `\textasciicircum` for `^`, and `\textbackslash` for `\` (writing `\\` would instead force a line break). Anything after an unescaped `%` becomes a comment, which is handy for leaving notes in your source but means a value such as a 90% win rate must be written `90\%`. The tilde has a second use outside `\textasciitilde`: on its own, `~` is a non-breaking space, explained in Section III.

When you need to show code or literal symbols without any LaTeX interpretation, use `\verb` for a short inline snippet. It works by picking a delimiter character that does not appear in your snippet and placing a copy on each side; the vertical bar is used throughout this template, but any unused symbol works (for example `\verb` with plus signs was used to typeset the bars you see here). For whole blocks, such as a snippet of game code see Section VIII.

A footnote adds a small note at the bottom of the page.¹ Write it with `\footnote{...}` placed *immediately* after the word it attaches to, with no space before it, so the raised marker sits tight against that word. LaTeX numbers footnotes automatically.

Web addresses should be set with `\url` (from the `url` package), which uses a monospace font and lets a long link break sensibly across lines: for example, the SDU site is `https://www.sdu.dk`. Do not type a bare URL as ordinary text, because its `~`, `%`, or `#` characters are special (see above) and would need escaping. In a report, a footnote and a URL often go together — cite a web resource in a footnote using `\footnote{\url{...}}`.

Lead Designer: Elev En	20%
Systems Designer: Elev To	20%
Lead Programmer: Elev Tre	20%
QA Lead: Elev Fire	20%
Art Director: Elev Fem	20%

¹Like this one — useful for an aside or a source that is not a formal reference.

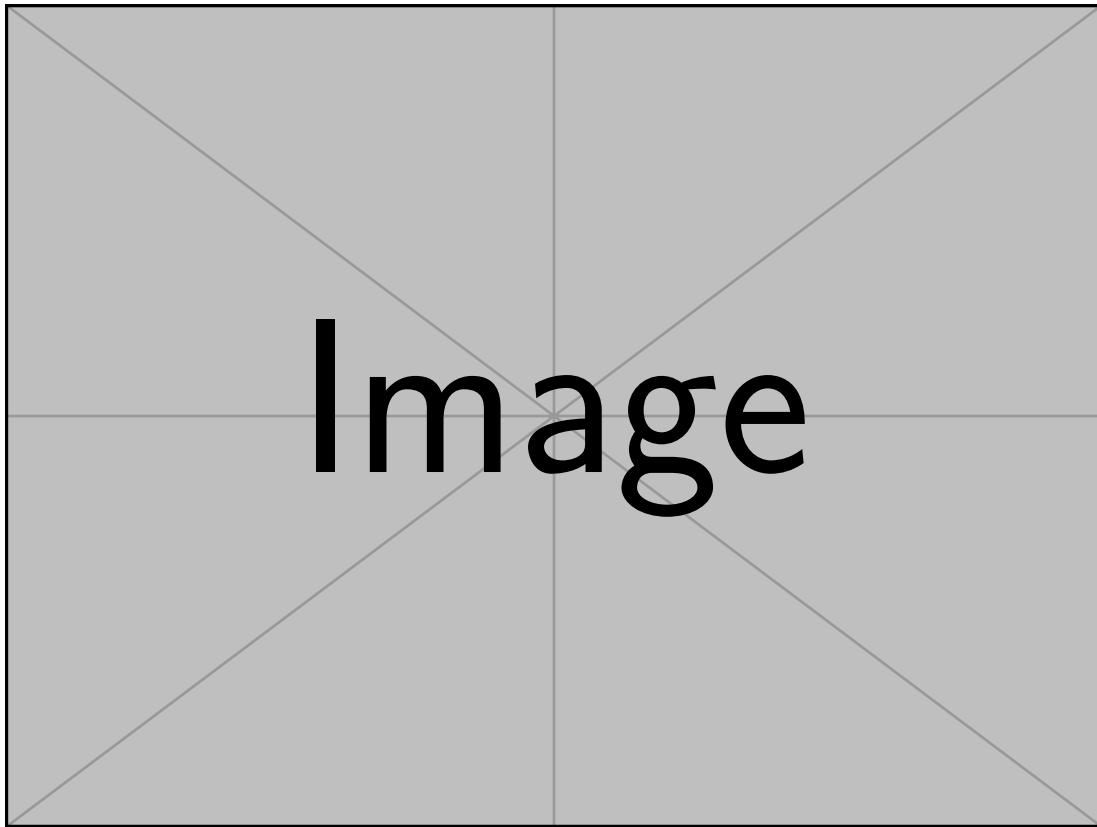


Fig. 1. A screenshot of the prototype's first playable level.

II. CITATIONS AND REFERENCES

References are managed using a separate BibTeX file (e.g., `references.bib`). Each source is one entry in that file: an entry type (such as `@book`, `@article`, or `@inproceedings`), a citation *key* you choose, and a set of fields describing the source. For example, the entry below defines a book with the key `tekinbas2003rules`:

```
@book{tekinbas2003rules,
  title={Rules of play: Game design
        fundamentals},
  author={Tekinbas, Katie Salen and
        Zimmerman, Eric},
  year={2003},
  publisher={MIT press}
}
```

To cite that source in your text, use the `\cite{}` command with its key: writing `\cite{tekinbas2003rules}` produces [1]. You never type the number yourself — BibTeX assigns and orders the numbers, formats each entry in the chosen style, and lists only the sources you actually cite.

A single reference can be cited as follows: [2]

Multiple references can be cited together: [2], [3]

When several sources support the same statement, they can be grouped into one citation. For example, research on player motivation and game design has been discussed extensively in the literature [2]–[4].

The bibliography is generated automatically at the end of the document. `\bibliographystyle` selects the formatting (here, the IEEE style) and `\bibliography` names your `.bib` file without the extension:

```
\bibliographystyle{IEEEtran}
\bibliography{references}
```

Note that BibTeX requires an extra build step: compile the document, run BibTeX, then compile twice more so the citation numbers settle. Most editors (Overleaf, TeXstudio, and others) do this for you automatically.

A. Quotation Marks and Quotes

LaTeX distinguishes opening from closing quotation marks, and the straight typewriter character " produces neither correctly. To get properly curved marks, type two backticks to open and two apostrophes to close: ``...'' gives “like this”, whereas "... " gives the wrong shape. The same applies to single quotes with `...`.

For a short quotation, keep it inline within your own sentence and cite the source. For a longer passage, set it apart using the `quote` environment, which indents the block so the reader can see at a glance that the words are not yours. Always attribute a block quote with a `\cite`:

Game design and authorship happen at many levels, and the fields of games research and development involve people from diverse creative and scholarly backgrounds. While it’s often necessary to focus on one area, everyone, regardless of discipline, will at some point need to consider issues outside that area: base mechanisms of game systems, the overarching design goals, or the desired experiential results of gameplay [4].

Use block quotes sparingly — a report should mostly be your own analysis, with quotations reserved for wording that matters.

III. SECTIONS AND SUBSECTIONS

Documents are organised into a hierarchy of headings. `\section` creates a top-level heading (numbered automatically), `\subsection` a level below it, and `\subsubsection` a level below that. You do not number them yourself; LaTeX handles the numbering and will renumber everything if you reorder or insert sections. You might use this structure to break your game down from a high-level concept into individual systems.

To refer to a heading elsewhere, give it a `\label` immediately after the heading, then use `\ref` to print its number. For example, this section was labelled with `\label{sec:sections}` and can be cited as Section III; the citation guidance is in Section II. Because LaTeX fills in the number, your cross-references stay correct even if you reorder the document. A useful convention is to prefix labels by type — `sec:`, `fig:`, `tab:`, `lst:`, `eq:` — so their purpose is obvious.

Notice the `~` in `Section~\ref{sec:sections}`. That tilde is a *non-breaking space*: it inserts an ordinary space but forbids a line break at that point, so the word “Section” can never be separated from its number across two lines. Always use `~` between a label word and its reference (`Figure~\ref{...}`, `Table~\ref{...}`), before a `\cite`, and in similar tight pairs such as “Section III” or “Elev En”.

A. Core Mechanics

A subsection covers a more specific topic within its parent section, such as the handful of core mechanics that define how your game plays.

1) *Enemy Behaviour*: A smaller heading can break a subsection down further — for example, the behaviour of a single enemy type within the game’s combat system.

IV. LISTS

Lists present a series of related items. LaTeX provides three built-in kinds, each an environment whose entries are marked with `\item`. Choose the kind by what the order of the items means: use a bullet list when order does not matter, a numbered list when it does (steps, rankings), and a description list when each item is a term being defined.

A. Bullet List

A bullet list uses the `itemize` environment, for items with no particular order, such as a game’s headline features:

- Local co-op multiplayer
- Procedurally generated levels
- Unlockable playable characters

B. Numbered List

A numbered list uses the `enumerate` environment, for ordered items such as the steps of a design iteration. The numbers are generated automatically:

- 1) Prototype the core mechanic
- 2) Playtest with target players
- 3) Iterate on the feedback



Fig. 2. The game's full world map, shown across both columns.

C. Description List

A description list uses the `description` environment. The bracketed text after each `\item` becomes the bold term being defined, which is useful for a glossary of game terms:

Sprite A 2D image drawn as part of the game world.

Hitbox The region used to detect collisions.

Tick A single fixed update of the game simulation.

Lists can also be nested inside one another, and the `enumitem` package (already loaded) lets you customise the labels and spacing if needed.

V. FIGURES

Images are placed with `\includegraphics` inside a `figure` environment, which turns the image into a numbered, captioned *float* that LaTeX positions for you. The optional argument in brackets is a *placement hint*: `t` (top of a page), `b` (bottom), `h` (here, approximately), and `p` (a page of floats). You may combine letters — the hints are tried in the order LaTeX prefers, not the order you write them.

For this template, prefer `[t]`: top-placed figures give the cleanest, most consistent layout in a two-column document, and it is the IEEE convention. If a figure will not sit where you want, or LaTeX pushes it far from its text, relax the hint rather than fight it: `[ht]` lets LaTeX try “here” before falling back to the top, `[h]` asks for roughly the current position, and adding `!` (as in `[ht!]` or `[!t]`) tells LaTeX to ignore its usual spacing limits and honour your hint more aggressively. Reach for these only when the default causes a real problem — overusing `h` tends to leave awkward gaps. Set the image size relative to the text with `width=0.8\linewidth` rather than in fixed centimetres, so it scales with the layout. Always give each figure a `\caption` and a `\label` so it can be referenced; place the `\label` after the `\caption`.

Figures can then be referenced by number, e.g. Figure 1 shows the level. Because floats move, always refer to them by `\ref` (with a non-breaking `~` before it) rather than by position (never “the screenshot below”).

A figure that needs the full width of the page, spanning both columns, uses the starred form `figure*` with `width=\textwidth` — useful for a wide world map or a full UI mock-up. Note that two-column floats can only appear at the top of a page, so they often move further from their source position than single-column figures do.

Figure 2 spans both columns in this way. This obviously has little to no effect in the one-column layout of the template.

VI. SUBFIGURES

Several related images can be grouped inside a single figure using `\subfloat` (from the `subfig` package). Each subfigure gets its own caption and its own `\label`, so you can reference either the whole figure or an individual part. The `\hfill` between them spreads them evenly across the column. This is ideal for showing how a level evolved between iterations.

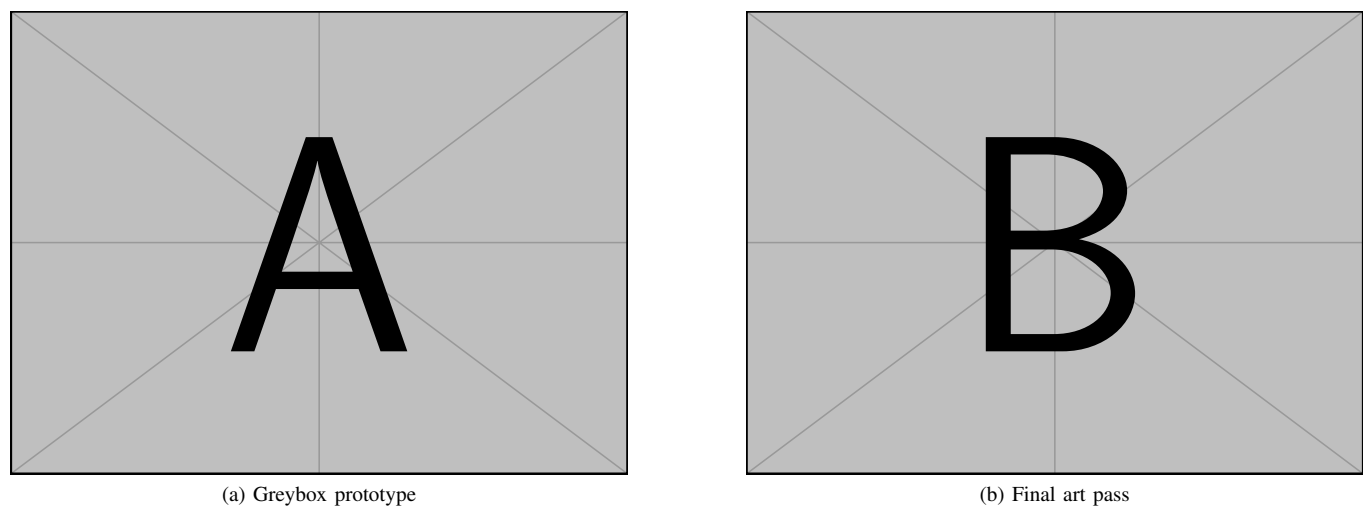


Fig. 3. Two iterations of the same level: early greybox and final art.

TABLE I
DEFAULT KEYBOARD CONTROLS

Action	Key
Move	W A S D
Jump	Space
Interact	E
Pause	Esc

TABLE II
PLAYTEST RESULTS BY AI VARIANT AND DIFFICULTY

AI Variant	Difficulty	Player Outcome		Rating ^a
		Win %	Avg. Time	
Patrol	Normal	72	4:15	3.8
	Hard	48	6:02	4.1
Flanker	Normal	55	5:30	4.4
	Hard	31	8:47	4.6

^aMean fun rating on a 1–5 scale ($n = 20$).

Figure 3 compares the two versions — the greybox stage is Figure 3a and the final art is Figure 3b. Note that each subfigure label resolves to a letter (a), (b), while the parent figure resolves to its number.

VII. TABLES

Tables are useful for presenting structured information. The simplest form is a few columns and rows inside a `tabular` environment, wrapped in a `table float` so it can carry a caption and be referenced. The argument to `tabular` declares one column each: `l`, `c`, and `r` give left-, centre-, and right-aligned columns; rows are separated by `\\`. In professional IEEE style, tables use *no vertical rules* and only a few horizontal ones. The `booktabs` package provides these: `\toprule` at the top, `\midrule` under the header, and `\bottomrule` at the bottom — and its manual advises against ever adding vertical lines. Note also that a table’s `\caption` goes *above* it (unlike a figure’s, which goes below). Table I shows a basic example.

For richer data, a table can group columns under a shared heading with `\multicolumn` and let a label span several rows with `\multirow`. A `\cmidrule` draws a partial rule under just the grouped columns. Table II summarises playtest results for two enemy AI variants across two difficulty settings.

Table II presents the results. Note how `\multicolumn{2}{c}{...}` merges two header cells into one, and `\cmidrule{1r}{3-4}` rules only the two columns under that grouped heading (the `(1r)` trims the rule slightly so it does not touch neighbouring columns). A `\midrule` separates the two AI variants. For a table too wide for one column, wrap it in a `table*` float so it spans both columns, exactly as with a full-width figure.

```

1 // Physics advances at a constant rate; rendering runs
2 // as fast as the platform allows.
3 const double dt = 1.0 / 60.0; // 60 physics updates / second
4 double accumulator = 0.0;
5 double previous = getCurrentTime();
6
7 while (running) {
8     double now = getCurrentTime();
9     accumulator += now - previous;
10    previous = now;
11
12    processInput();
13
14    // Consume fixed-size steps until we have caught up.
15    while (accumulator >= dt) {
16        updatePhysics(dt);
17        accumulator -= dt;
18    }
19
20    // Blend factor for interpolating between physics states.
21    double alpha = accumulator / dt;
22    render(alpha);
23 }

```

Fig. 4. Fixed-timestep game loop.

VIII. CODE LISTINGS

Source code is presented with the `lstlisting` environment, which preserves indentation and adds syntax highlighting. Placing it inside a figure gives it a caption, a label, and the same spacing as any other figure. The listing below shows a *fixed-timestep game loop*: physics is advanced in constant-size steps so that simulation behaviour is independent of frame rate, while rendering is interpolated to stay smooth on faster displays.

Listing 4 decouples the simulation rate from the render rate, a common requirement for deterministic gameplay and networked physics. Change the `language=` key in the `\lstset` block near the top of the file to highlight a different language, such as C# for Unity.

IX. ALGORITHMS

Higher-level logic is often clearer as pseudocode than as source code. The `algorithm` and `algorithmic` environments are used for this: the `algorithm` wrapper provides the caption and number (like a float), while `algorithmic` formats the steps. Commands such as `\STATE`, `\IF`/`\ENDIF`, `\WHILE`/`\ENDWHILE`, and `\FORALL` set the keywords in the conventional style, and the optional `[1]` numbers the lines. Algorithm 1 describes A^* *pathfinding* over a navigation grid, a standard technique for moving enemies around obstacles.

The heuristic $h(n)$ estimates the remaining cost to the goal (for a tile grid, Manhattan or Euclidean distance is typical). As long as h never overestimates, A^* is guaranteed to return a shortest path for the enemy to follow.

X. MATHEMATICS

Mathematics is written in *math mode*. For a short formula inside a sentence, use inline math delimited by dollar signs: the damage after a 25% armour reduction is $d' = 0.75d$. For a formula you want set apart on its own line and automatically numbered, use the `equation` environment:

$$d' = d \left(1 - \frac{a}{a + k} \right) \quad (1)$$

Equation 1 is a common armour-damage formula, where a is armour and k a tuning constant, and `\label` lets you refer back to it by number. If you want a displayed formula *without* a number, use `\[ ... \]` (or the `equation*` environment) instead.

Superscripts are written with `^` and subscripts with `_`. Each applies to the single character that follows, so wrap anything longer in braces: 2^n gives 2^n but $2^{\{n-1\}}$ is needed for 2^{n-1} , and $v_{\{max\}}$ gives v_{max} . The two can be combined on the same symbol, as in x_i^2 for x_i^2 . This is how you write things like a player's health h_i on frame i , or a score that grows as 2^n with the combo count n .

Beyond these, a few constructs come up constantly. Fractions use `\frac`, Greek letters have named commands such as `\alpha` and `\theta`, and operators like $\sum$, $\sqrt$, and $\leq$ have their own commands too. For instance, interpolating a character between two waypoints and averaging a score over n playtests can be written as Equation 2:

Algorithm 1 A* Pathfinding over a Navigation Grid

Require: start node s , goal node t , admissible heuristic $h(n)$
Ensure: a shortest path from s to t , or none if t is unreachable

```

1:  $open \leftarrow \{s\}$  {priority queue ordered by  $fScore$ }
2:  $gScore[s] \leftarrow 0$ 
3:  $fScore[s] \leftarrow h(s)$ 
4: while  $open \neq \emptyset$  do
5:    $current \leftarrow \arg \min_{n \in open} fScore[n]$ 
6:   if  $current = t$  then
7:     return RECONSTRUCT( $cameFrom, current$ )
8:   end if
9:   remove  $current$  from  $open$ 
10:  for all neighbours  $n$  of  $current$  do
11:     $tentative \leftarrow gScore[current] + \text{COST}(current, n)$ 
12:    if  $tentative < gScore[n]$  then
13:       $cameFrom[n] \leftarrow current$ 
14:       $gScore[n] \leftarrow tentative$ 
15:       $fScore[n] \leftarrow tentative + h(n)$ 
16:      if  $n \notin open$  then
17:        add  $n$  to  $open$ 
18:      end if
19:    end if
20:  end for
21: end while
22: return none

```

$$p(t) = (1 - t)p_0 + tp_1, \quad \bar{s} = \frac{1}{n} \sum_{i=1}^n s_i \quad (2)$$

Note that spaces in the source are ignored in math mode, so LaTeX handles the spacing for you; use `\,` or `\quad` only when you want to fine-tune it. Remember also that symbols such as `^`, `_`, and Greek letters work *only* inside math mode — writing them in ordinary text will produce an error or the wrong output.

XI. CONCLUSION

Summarise the main outcomes of your game project: what you built, how it played, and what your playtests showed. Keep it short — the detail belongs in the sections above.

ACKNOWLEDGMENTS

The main use of the Acknowledgments section is to declare the usage of generative AI in the project (e.g. assets, code) and the writing of the report.

REFERENCES

- [1] K. S. Tekinbas and E. Zimmerman, *Rules of play: Game design fundamentals*. MIT press, 2003.
- [2] S. Rigby and R. M. Ryan, *Immersion and Presence*. Bloomsbury Publishing USA, 2011, ch. 5, pp. 81–96.
- [3] K. Salen and E. Zimmerman, “Game design and meaningful play,” *Handbook of computer game studies*, vol. 59, no. 79, pp. 2–1, 2005.
- [4] R. Hunicke, M. LeBlanc, R. Zubek et al., “Mda: A formal approach to game design and game research,” in *Proceedings of the AAAI Workshop on Challenges in Game AI*, vol. 4, no. 1. San Jose, CA, 2004, p. 1722.

APPENDIX A

COMPILING THIS TEMPLATE

This appendix explains how to build the template on Windows using **VS Code** and **MiKTeX**. Follow the steps in order.

- 1) Download MiKTeX from <https://miktex.org/download> and run the installer. You can accept the defaults.
- 2) When asked to install missing packages on-the-fly, choose **Yes**. This lets MiKTeX fetch packages such as booktabs automatically.
- 3) Open the MiKTeX Console once and apply any updates it offers.
- 4) Download VS Code from <https://code.visualstudio.com> and install it.
- 5) In VS Code, open the Extensions view (Ctrl+Shift+X), search for **LaTeX Workshop**, and install it.
- 6) Open the Command Palette (Ctrl+Shift+P) and run *Preferences: Open User Settings (JSON)*.
- 7) Type *latex recipe* in the search box. Set *Latex-workshop Latex: Recipe.default* to `pdflatex -> bibtex -> pdflatex * 2`.
- 8) Unzip the template with `main.tex` and `references.bib` in one folder, and open that folder with *File* → *Open Folder*.
- 9) Open `main.tex` and save it (Ctrl+S). The PDF builds automatically; you can also press Ctrl+Alt+B.
- 10) The first build is slow. If MiKTeX asks to install packages, click **Install**. This happens only once.
- 11) Press Ctrl+Alt+V to view the PDF.

If you get errors regarding `pdflatex`, Open the Command Palette (Ctrl+Shift+P) and run *Preferences: Open User Settings (JSON)*. Type *latex recipe* in the search box. Click on Edit in `settings.json` and paste the following block; overriding the previous setup.

```
{
  "workbench.secondarySideBar.defaultVisibility": "hidden",
  "latex-workshop.latex.tools": [
    {
      "name": "pdflatex",
      "command": "pdflatex",
      "args": [
        "-synctex=1",
        "-interaction=nonstopmode",
        "-file-line-error",
        "%DOC%"
      ]
    },
    {
      "name": "bibtex",
      "command": "bibtex",
      "args": ["%DOCFILE%"]
    }
  ],
  "latex-workshop.latex.recipes": [
    {
      "name": "pdflatex -> bibtex -> pdflatex x2",
      "tools": ["pdflatex", "bibtex", "pdflatex", "pdflatex"]
    }
  ],
  "latex-workshop.latex.recipe.default": "first"
}
```

If citations show as “[?]”, build once more. If a build error names a missing package, let MiKTeX install it and build again. If nothing happens on save, check that you opened the folder (not just the file) and that LaTeX Workshop is enabled.

APPENDIX B

FULL PLAYTEST QUESTIONNAIRE

Appendices hold supporting material that would interrupt the main text — raw data, long tables, or full questionnaires. For a single appendix use `\appendix`; for several, use `\appendices` and begin each one with `\section`, which LaTeX labels “Appendix A”, “Appendix B”, and so on. Appendices can be labelled and referenced just like sections, so this one is Appendix B. It would contain the complete list of questions given to playtesters after each session, including the fun-rating scale referenced in Table II.

APPENDIX C

EXTENDED CONTROLS REFERENCE

This appendix (Appendix C) would list the full keyboard and gamepad control scheme, including the secondary and debug bindings omitted from the summary in Table I.