Elision Based Text Zooming

Sam Davis*

University of British Columbia

Abstract

Scrolling through text documents is a cumbersome and ineffective means of getting context and overview. I present a superior technique that augments scrolling by using elision to simulate zooming. This allows smooth, rapid transitions between overview and detail and effectively supports the task of recovering lost context when reading or navigating through a document. I describe an implementation of the technique for a popular web browser.

CR Categories and Subject Descriptors:
Additional Keywords:

1 Introduction

While scrolling might be thought of primarily as a method of navigation, in the domain of text documents, it is also typically the only method of providing overview and focus+context. Where an overview is not explicitly provided as part of the document, or where an overview is provided but is not sufficiently detailed, users often scroll through the entire document in an attempt to understand its overall structure. When reading the document, scrolling to nearby locations is used to provide context.

Unfortunately, scrolling is poorly suited to these tasks. For all but the shortest of text documents, scrolling from beginning to end must either take unreasonably long or cause text to rapidly fly by in a blur that conveys little or no information. Attempting to compromise between these two extremes by scrolling in increments is still time consuming, is cumbersome, and does not make the important parts of the document easy to find – the scrolling process is not sufficiently more likely to pause when something important is in view than when something unimportant is in view.

Scrolling also hampers the process of getting context, because the time and cognitive effort expended in moving back and forth between related parts of the document via the scrollbar may cause the user to forget what they were looking for, or why they were looking for it.

While scrolling is adequate for navigation, it is still not easy to navigate to an item of interest, unless one knows exactly how far it is from the beginning or end of the document. The ability to search the text for keywords can be helpful, but it fails when the user does not know the precise words used in the interesting part of the document, and when the search yields too many results. Personal experience suggests that many users tend not to use text search features, perhaps because they often require using both the keyboard and the mouse, and take too long to return results. Another reason may be that there is generally no automatic way to return to the location that was in view prior to initiating the search, so the user gets lost.

This paper explores the application of elision to simulate zooming, in an attempt to support context and overview more naturally.

The paper is structured as follows. Section 2 provides an overview of related work. Section 3 describes a technique for simulating zooming using elision, and Section 4 discusses the details of an implementation of that technique. Scenarios of use are presented in Section 5, and Section 6 concludes.

2 Related Work

Prior work on scrolling text focuses on shrinking the text or changing the document representation, rather than elision. Speed-Dependent Automatic Zooming [6] reduces text size in proportion to scrolling speed, rather than providing an explicit zooming control, but as with any method that relies on shrinking text, it suffers from legibility problems. The OrthoZoom Scroller [3] uses the two degrees of freedom of the mouse to separately control zooming and scrolling, and relies on the existence of a multi-scale table of contents to overcome the problem of illegible text.

Previous uses of text elision include source code folding, where blocks of code are selectively hidden. In the simplest versions, the user must manually indicate which blocks to hide, as in the Eclipse IDE [5], resulting in a feature of debatable usefulness. The JSEclipse plug-in [2] goes one step further by automatically hiding commented blocks of code, significantly increasing the usefulness of the code folding feature. Jakobsen and Hornbaek [7] describe an Eclipse plug-in that presents a fisheye view of source code, where the main source code view is surrounded by a context area which displays other lines of code deemed relevant based on semantic relations extracted from the code. Mylar [8], another Eclipse plug-in, determines which elements of the source code to hide by using a recorded history of user actions to determine which elements are most relevant to the task the user is performing.

Fluid Documents [4] apply elision to text other than source code by using interaction to indicate when hidden information should be shown. Because this hidden information consists of definitions and explanations that are not part of the core document, Fluid Documents could be more accurately described as adding annotations to a document rather than using elision to reduce its size.

The Adobe Reader [1] attempts to solve the problem of going back to previously seen parts of a text by providing back and forward buttons, but the number of views remembered is so large that the buttons are almost useless. It also suffers from the "forking" problem common to web browsers, where going back and then navigating to another location erases the forward history. The approach could be combined with the zooming described in this paper by using the locations that were zoomed on as the views to remember.

3 Elision Based Text Zooming

Rather than shrinking text to the point of illegibility, elision based text zooming produces a zoomed out representation of a document by hiding the end of each paragraph. As the level of zoom is increased or decreased, the amount of each paragraph that is hidden is increased or decreased respectively, in a way that can depend on the size of the paragraph. The result is a smooth

* sddavis@cs.ubc.ca

transition from the lowest level view (the document itself without any zooming) to successively higher level views, with the highest level being only the headings (if present) or the beginnings of each paragraph.

This technique facilitates rapid scrolling through a long document by reducing its length. It also allows one to see an overview of the document by reducing it to only its headings or the beginning of each paragraph, and to quickly and smoothly transition from overview to details or vice versa without getting lost. When reading a document at the lowest level, these quick transitions can be used to recover forgotten context. For documents with little explicit structure (i.e. headings), the effectiveness of the technique rests on the assumption that people rely primarily on the beginning of a paragraph to make a quick determination about its relevance.

Screenshots are shown in Figures 5 through 8.

3.1 Controls

Zooming is controlled using the horizontal axis of the mouse. When the right mouse button is held down, a zooming widget is displayed. Moving the mouse to the left increases the zoom (causes text to be hidden), and moving it to the right reduces the zoom.

The zooming widget (Figure 1) is displayed on top of the document, at the location of the mouse when the button was first clicked. Designed to be easily seen but to have a low visual footprint, the widget consists mainly of a red, one pixel wide horizontal line extending about half the width of the screen. At the right end of the line, a small glyph indicates the zoom direction and also marks the position at which the zoom is zero (i.e. nothing is hidden). The red line is intended to draw the eye to the text underneath without obscuring it.

A small circular thumb which follows the mouse indicates the current zoom. The thumb is prevented from moving beyond the right edge of the line, as the widget does not currently support zooming in beyond 100%, but it is allowed to move as far to the right as the user desires, as the maximum possible zoom depends on the document. A probable improvement would be to set the length of the line to correspond to this maximum possible zoom and prevent the thumb from exceeding it.

3.2 Elision Increment

The original intent was to hide text a line or lines at a time from each paragraph. However, as an approximation to this, text was hidden twenty words at a time and the result was that zooming was too choppy – it was very difficult to track the text as it moved because of the large sudden jumps in position when every paragraph in the document lost a line at the same time. Instead, it was decided to hide text in groups of six words. The number six was found by trial and observation to provide a good balance between smoothness and performance of the implementation.

In theory, the result of ignoring line breaks is a sub-optimal use of space in that a line that shows one word or five words takes up as much vertical space as a line that shows twenty. However, it may be that the empty space is actually more valuable than the text which could fill it. Combined with the less uniform line lengths that it produces, the empty space might actually help the user to remain oriented in the document. It would be interesting to compare the usability against an implementation which attempts to pack more text onto the screen by hiding entire lines. This could be made smoother by hiding lines from only a fraction of the paragraphs in each zooming step.

3.3 Elision Speed

The speed at which text is elided is determined differently when zooming in than when zooming out. When zooming in, all paragraphs are zoomed at the same fixed rate. In contrast, when zooming out, each paragraph is zoomed at a speed proportional to its length. For a paragraph of length `n` (in words), the number of words to be hidden for a given zooming step is proportional to the size of the zooming step (determined by the horizontal distance the mouse was moved) and to a scale factor

```
f = 0.5 + n / 90.
```

This value of `f` was arrived at through experimentation and is not claimed to be optimal, however it works well in practice.

The motivation behind allowing the paragraph size to affect the rate at which it zooms was to prevent longer paragraphs from hiding shorter ones when zooming out. If all paragraphs were zoomed out at the same speed, the smaller ones would all but disappear while the larger ones continued to take up significant screen space, making it difficult to obtain an overview. While the approach taken here means that larger quantities of text will be hidden at once, continuity is maintained because the focal point of the zoom is fixed on the screen, as explained in Section 3.4.

Zooming in at rates proportional to paragraph length was tried and found to be disorienting because of the sudden appearance of large amounts of text. Zooming out proportionally and then zooming in at a fixed rate causes paragraphs of different lengths to be quickly made about the same length, after which they remain that way until the zoom is reduced to zero. At that point, the larger paragraphs suddenly expand to expose their full contents. To my surprise, this sudden change is not hard to follow, probably because a significant amount of the text displayed on the screen when almost fully zoomed in remains roughly in place, and also because the change happens at a specific point (just as the user zooms all the way in), rather than happening continuously as it would with proportional zooming in.

3.4 Centering The Zoom

As a consequence of the fact that paragraph lengths vary (and not as a result of the method of choosing the speed of elision), the location of each paragraph as a percentage of the displayed length of the document changes while zooming. The question of how to center the zoom therefore needs to be answered, as simply keeping the view centered over the part of the document expressed as a percentage of the length results in an uncontrollable scrolling, often oscillating wildly up and down.

The solution adopted was to fix the position of the document element at the location initially clicked (i.e. independent of subsequent vertical motion of the mouse during zooming). The effect is that text collapses towards the zooming widget when zooming out and expands away from it when zooming in. As mentioned above, the zooming widget draws attention to the text underneath it, and this helps to prevent disorientation by keeping the user's focus on the focal point of the zoom.

As a refinement, rather than fixing the element on which the user clicked, the element fixed is the one whose top is vertically closest to the location initially clicked. This serves two purposes. Firstly, if the user clicks in the document margin, there will be no element directly under the mouse, and secondly, if the user clicks on the end of a paragraph, it is better to fix the beginning of the next paragraph than the one actually clicked, as it will be closer to the mouse (and therefore to the zooming widget).[1] A further benefit to this approach is that, if the focal point of the zoom becomes hidden (as can happen with images, for example), the

1 It does not make sense to fix the end of the paragraph clicked as this will almost immediately be hidden.

focal point can be shifted to be the element which was next closest to the location clicked and is still visible. This element will generally have moved to be very close to the original focal point, so no sudden jump or discontinuity in the zooming will be apparent to the user.

The net effect of all this is to give the user control over the focal point of zooming. The user can easily zoom in on a particular part of the document simply by pointing the mouse at it and zooming in. The facilitates a very natural interaction paradigm where the user first zooms out to a high level overview of the document, selects a different element (possibly by scrolling the overview), an zooms in on it, moving smoothly and rapidly between different sections of the document.

3.5 Indicating Where Elision Occurs

Each partially hidden paragraph displays a glyph to indicate the location and quantity of hidden text. This glyph is a small rectangle containing an ellipsis (see Figure 2) and is similar in appearance to that used by the Eclipse IDE's [5] source code folding feature to indicate where a block of code has been hidden. The glyph used here has two important differences, both motivated by personal experience with Eclipse. Firstly, it is colored dark red, in contrast to Eclipse's glyph which is light gray. This significantly increases the visual salience of the glyph, making it easy to notice where text has been hidden. In Eclipse, it is easy to miss the less obvious gray ellipsis glyph, although the problem is mitigated to some extent by the visual structure of source code. This is because the unit of hiding in source code folding is a (usually indented) code block, surrounded by braces, so the absence of the hidden block between the braces is more apparent at a glance, without any indicator, as compared to text which lacks the visual structure, and whose unit of elision is based on words. The second difference is that the ellipsis glyph used in here also indicates the amount of text which has been hidden from the paragraph. This is done by making the width of the rectangle and the number of dots in the ellipsis proportional to the amount of text which has been hidden. Each dot in the glyph corresponds to roughly one line of hidden text. This visual encoding should be readily understood because it uses a familiar symbol (the ellipsis) with a well-established meaning. It is also compact: an ellipsis glyph the size of a single 5 character word will represent about 6 to 12 lines of hidden text, depending on the paragraph width. The glyph thus takes up about 1/120 as much space as the text it represents, a zoom factor far greater than could be achieved by simply reducing the font size.

3.6 Details on Demand

To make the zoomed out representation of a document more informative, details on demand is incorporated in two ways. Simply clicking on a zoomed out paragraph immediately zooms in on just that paragraph. Subsequent zoom operations continue to effect that paragraph, but it remains zoomed in relative to other paragraphs until the document has been fully zoomed in. This provides a degree of focus+context by allowing the user to see some parts of the document in detail while still seeing some of the surrounding context. Clicking a paragraph also draws a permanent border around it as shown in Figure 4 (compare with Figure 2), to allow the user to easily find it again.

The other way in which details on demand is incorporated is via the ellipsis glyphs. Moving the mouse over an ellipsis glyph pops up a gold box containing a compact representation of the complete paragraph, including the hidden text (see Figure 3). This representation of the paragraph uses a smaller font (hence the compactness) but otherwise appears the same as the original paragraph, including paragraph width, formatting, images, etc.. An important feature is that the first word which was hidden (that is, the word which follows the last visible word in the elided representation) is highlighted in red and has a dashed, blue border drawn around it. This coloring immediately draws the eye to the beginning of the elided text, making it possible to read the elided representation to the end of the visible text, and then move the mouse over the ellipsis glyph and seamlessly continue reading the compact representation from the next word without missing a beat.

4 Implementation

4.1 Tools Used

The techniques described in this paper were implemented as an extension to the Firefox web browser using JavaScript and CSS. The implementation is therefore cross-platform and can be used by real users on real web pages, without requiring them to use a specialized tool.

Firefox provides access to the HTML document tree through the DOM (Document Object Model) interface, which represents the hierarchical structure of the document as a tree whose root is the entire document. Each HTML tag corresponds to a node in the tree, and the plain text between two adjacent tags is represented as one or more (sibling) text nodes. Typically, what appears as a paragraph in the rendered page is a node in the DOM, with child nodes corresponding to the text, formatting, links, and other sub-structures contained in the paragraph. Most of the implementation of this project involved performing transformations and searches on the DOM. Even the visual feedback such as the zooming widget and the details on demand pop-up was implemented by inserting nodes into the DOM.

The DOM representation is not ideal for performing advanced document transformations in that the hierarchy represents both structure and formatting. The result is that what appears as a hierarchical structure in the document is not necessarily hierarchical in the DOM. This turned out to be a more prevalent phenomenon than I had expected. However, given the time constraints, I do not believe it would have been feasible to create my own document representation and renderer. Working around the limitations of the DOM was not easy but allowed me to take advantage of the Gecko rendering engine built into Firefox and a well-established format with a boundless supply of documents for testing.

In addition to representing the document, the DOM also provides an interface to the browser itself. This interface was used to install the event handlers that drive the DOM transformations.

4.2 Preprocessing

When a page is loaded, a substantial amount of preprocessing is done to prepare the page for zooming. Each node is tagged with various properties indicating, for example, whether it should be treated as a paragraph and whether it is allowed to be elided itself. For instance, headings are never hidden, and this property is recursive, that is, no child of a heading will be hidden. This is one of many cases where the hierarchy of the DOM differs from the hierarchy perceived by the user.[2]

For simplicity, the initial implementation split strings on spaces each time operations were to be done at the word level. Not surprisingly, this proved to be too slow. A preprocessing step was

2 A more natural representation of the document would make the section identified by a heading be the child of that heading, but the DOM represents the section denoted by a heading as a sibling of the heading (or even of an ancestor of the heading).

added which splits each text node up into multiple text nodes, each containing (at most) six words. This allows the rest of the code to deal mostly with nodes in the tree rather than with strings. In addition, each node is also assigned a count of the number of words contained in all of its descendants. These word counts are updated as zooming is performed and are used to decide which text to hide, and how much.

The final preprocessing step actually replaces the entire document tree with a duplicate of itself. The original tree is kept in memory and hereafter is neither structurally modified nor directly displayed. Each node in the duplicate (displayed) tree has a pointer back to the corresponding node in the original (not displayed) tree. When zooming in to show nodes that have been hidden, these pointers are used to retrieve the hidden nodes from the original tree (by creating duplicates of the nodes, inserting them into the duplicate tree, and giving them pointers back to the original tree). When zooming out, the nodes to be hidden are simply removed from the document tree. The original tree is also used to recover properties of nodes have changed due to zooming, even though the nodes are not themselves hidden. For example, it is sometimes necessary to compare the current and original word counts of a node, or to know the original relative position of a node in the document.

4.3 Technical Limitations

Currently, the elision based text zooming tool works on a substantial percentage of the web pages tried, but also fails (does not produce a good zoomed out representation, performs too slowly, or exhibits bugs) on a substantial percentage. The bugs and performance problems are simply the result of time pressure during the implementation phase, the goal of which was primarily to explore the possible techniques as fully as time permitted, rather than to produce a commercially viable tool. There are consequently numerous inefficiencies in the code and deliberately unsupported browser features (for example, the tool does not interact perfectly with the Firefox browser's ability to open multiple pages in different tabs within the same window). I think that these problems would be straightforward to fix, given time.

The pages for which the tool produces a poor zoomed out representation appear to be primarily news articles which make heavy use of HTML's <br> tag to delimit paragraphs (in fact, it is mostly news articles which exhibit performance problems as well). Problems arise here and in other cases where the DOM representation diverges from the apparent structure of the document. In this case, a pair of <br> tags insert line breaks to create a blank line, thus giving the appearance of a paragraph without creating a node in the HTML document tree to represent it. It is worth pointing out that this issue is not due to any fundamental limitation in the zooming technique, it is simply an artifact of the implementation (one could argue that it is a flaw in HTML, or the way in which some developers use it). These issues could be resolved by augmenting the DOM with extra information that better represents the visual structure of the document. While the prototype-based nature of Javascript makes this sort of augmentation relatively easy, a really mature implementation of the technique would have to give careful individual treatment to a great many of these special cases.

5 Scenarios of Use

5.1 Scenario 1: Overview

The user is faced with a long web page with no table of contents and wants to understand the overall structure of it. He zooms out to produce an overview of the document and is able to immediately see the headings and sub-headings and their layout on the page. He then zooms in on an interesting heading and reads that section of the document.

5.2 Scenario 2: Search

The user is reading a document and remembers that the author has coined a term to describe something, which he believes is related to the part he is reading now, but he cannot remember the exact phrase. Holding down the mouse button to bring up the zooming widget, he zooms out of the document until only the first two lines of each paragraph are visible. This results in a much shorter document and he is able to quickly find a paragraph that looks relevant. Releasing the button, he moves the mouse to the interesting paragraph, and then again holds the button down and moves the mouse to zoom part of the way in on that paragraph, and realizes that it does not contain the definition he is looking for. However, he sees that the following paragraph looks very promising and clicks on it to show it in full. He finds the definition he is looking for within, and then uses the mouse to quickly zoom out and scroll back to the place he left off reading. Subsequently, he wishes to return to the definition and is able to quickly find it because a box has been drawn around it (as a result of having been clicked on).

6 Conclusion

This paper presented a technique for simulating zooming of text documents by elision and described its implementation as an extension to a popular web browser. The technique reduces user reliance on scrolling by serving as a navigation aid and also by providing much better support for overview and for focus+context. Users are able to move smoothly and rapidly from the lowest level view of a document to a higher level view to regain lost context when reading or to see an overview of the document.

Glyphs are used to indicate where text has been elided and details on demand is used to provide easy access to the hidden text. A method for adjusting the relative speed of zooming on a per-paragraph basis was described, as was a method for maintaining a coherent view during the zooming process.

While a formal user study was not conducted, the response of those who witnessed a demonstration of the tool was generally positive and enthusiastic. The technique appears to be useful, but would benefit from further study and fine tuning.

References

[1] Adobe. Adobe Reader. http://www.adobe.com/products/acrobat/readstep2.html. Dec. 2006.

[2] Adobe Labs. JSEclipse. http://labs.adobe.com/technologies/jseclipse/. Dec. 2006.

[3] C Appert, J. Fekete. OrthoZoom Scroller: 1D Multi-Scale Navigation. Proc. SIGCHI 06, pp 21-30. 2006.

[4] B. Chang, J. D. Mackinlay, P. T. Zellweger. Fluidly Revealing Information in Fluid Documents. AAAI Smart Graphics Spring Symposium, 2000.

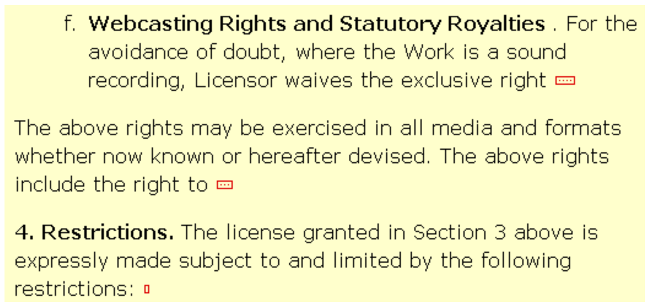
[5] Eclipse Foundation. Eclipse. http://www.eclipse.org/. Dec. 2006.

[6] T. Igarashi, K. Hinckley. Speed-Dependent Automatic Zooming for Browsing Large Documents. Proc. UIST'00, pp. 139-148. 2000.

[7] M. R. Jakobsen, K. Hornbæk. Evaluating a Fisheye View of Source Code. CHI, 2006.

[8] M. Kersten, G. C. Murphy. Mylar: a degree-of-interest model for IDEs. AOSD, 2005.

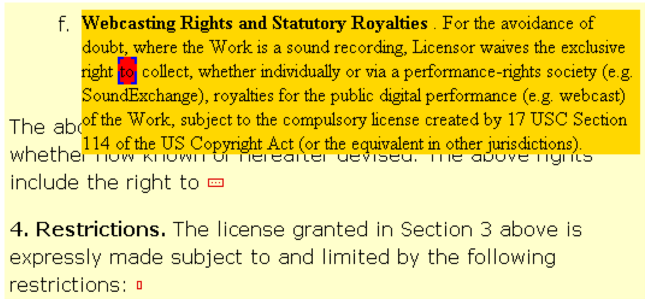
Figure 1


f. **Webcasting Rights and Statutory Royalties** . For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

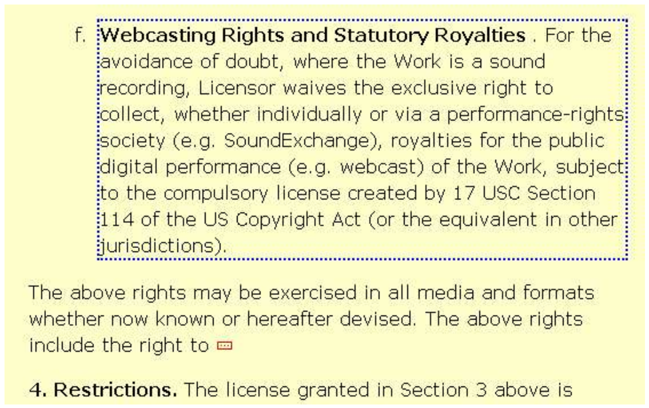
Figure 2


f. **Webcasting Rights and Statutory Royalties** . For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The abo
whether now known or hereafter devised. The above rights
include the right to

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

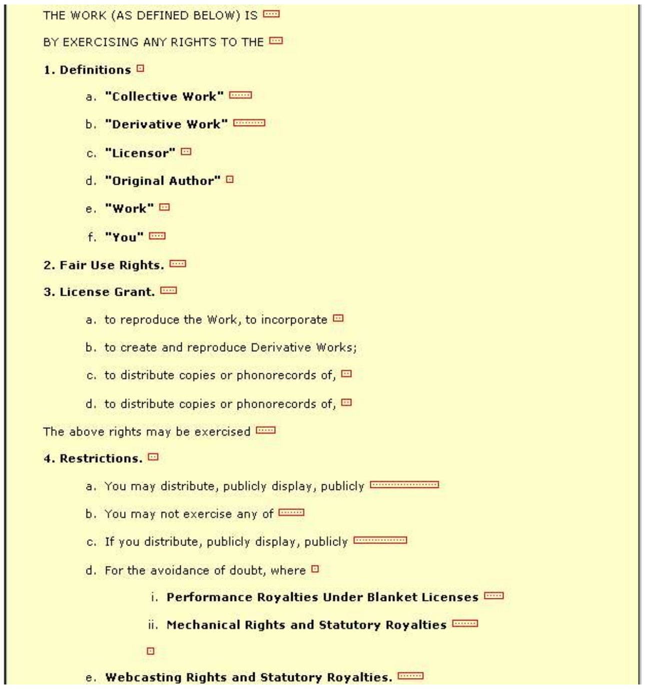
Figure 3


f. **Webcasting Rights and Statutory Royalties** . For the avoidance of doubt, where the Work is a sound recording, Licensor waives the exclusive right to collect, whether individually or via a performance-rights society (e.g. SoundExchange), royalties for the public digital performance (e.g. webcast) of the Work, subject to the compulsory license created by 17 USC Section 114 of the US Copyright Act (or the equivalent in other jurisdictions).

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to

4. Restrictions. The license granted in Section 3 above is

Figure 4


THE WORK (AS DEFINED BELOW) IS

BY EXERCISING ANY RIGHTS TO THE

1. Definitions

a. **"Collective Work"**

b. **"Derivative Work"**

c. **"Licensor"**

d. **"Original Author"**

e. **"Work"**

f. **"You"**

2. Fair Use Rights.

3. License Grant.

a. to reproduce the Work, to incorporate

b. to create and reproduce Derivative Works;

c. to distribute copies or phonorecords of,

d. to distribute copies or phonorecords of,

The above rights may be exercised

4. Restrictions.

a. You may distribute, publicly display, publicly

b. You may not exercise any of

c. If you distribute, publicly display, publicly

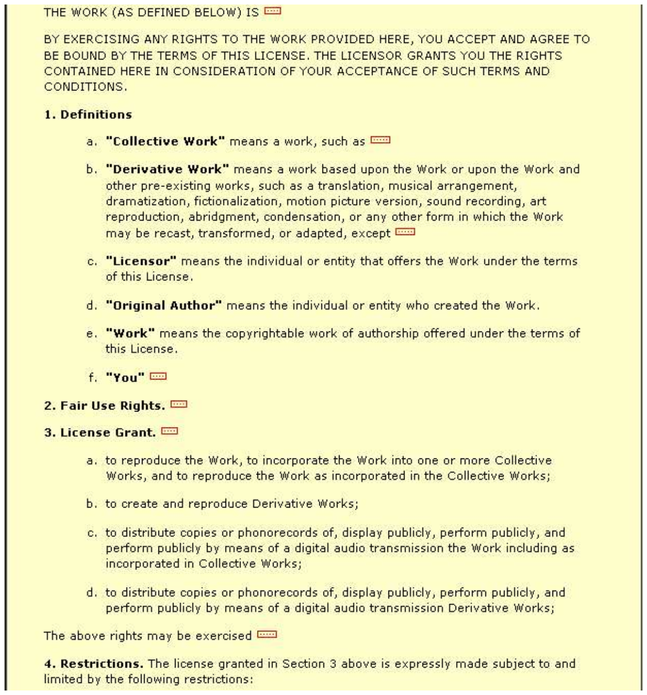
d. For the avoidance of doubt, where

i. **Performance Royalties Under Blanket Licenses**

ii. **Mechanical Rights and Statutory Royalties**

e. **Webcasting Rights and Statutory Royalties.**

Figure 5



THE WORK (AS DEFINED BELOW) IS

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Collective Work"** means a work, such as

b. **"Derivative Work"** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except

c. **"Licensor"** means the individual or entity that offers the Work under the terms of this License.

d. **"Original Author"** means the individual or entity who created the Work.

e. **"Work"** means the copyrightable work of authorship offered under the terms of this License.

f. **"You"**

2. Fair Use Rights.

3. License Grant.

a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

b. to create and reproduce Derivative Works;

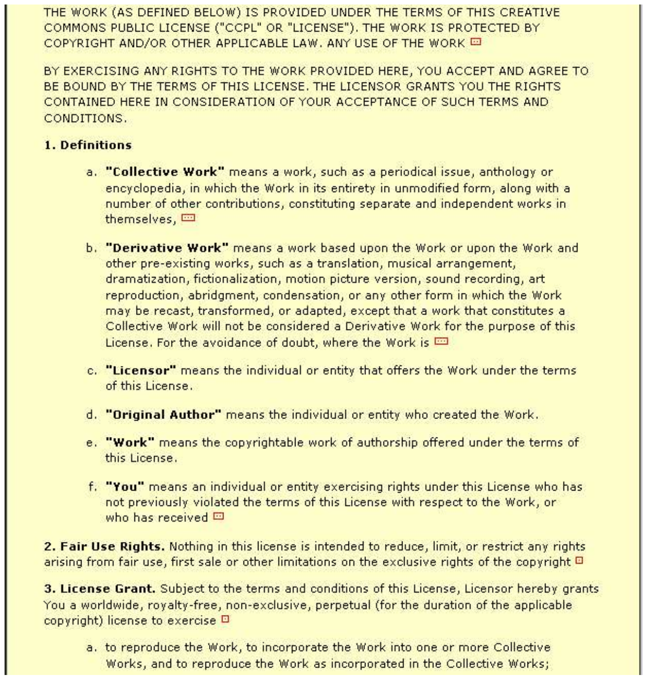
c. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

d. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission Derivative Works;

The above rights may be exercised

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

Figure 6



THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Collective Work"** means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves,

b. **"Derivative Work"** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is

c. **"Licensor"** means the individual or entity that offers the Work under the terms of this License.

d. **"Original Author"** means the individual or entity who created the Work.

e. **"Work"** means the copyrightable work of authorship offered under the terms of this License.

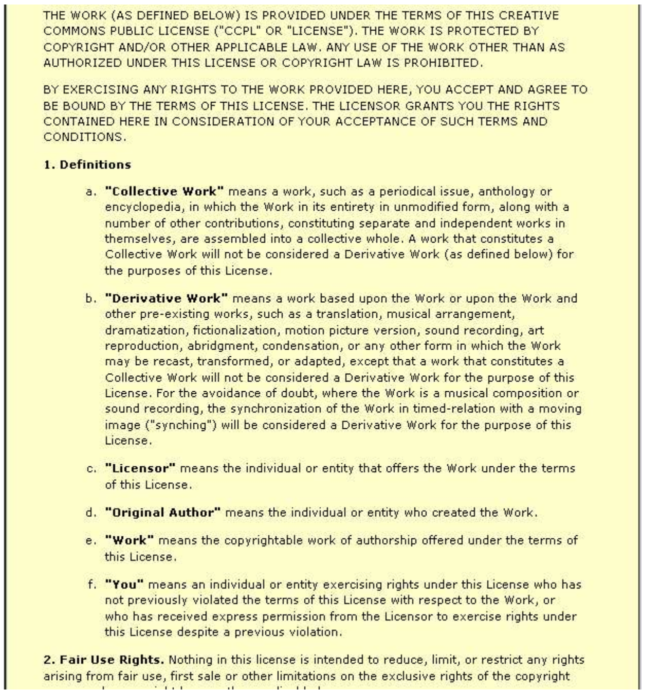
f. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise

a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

Figure 7



THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Collective Work"** means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.

b. **"Derivative Work"** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License. For the avoidance of doubt, where the Work is a musical composition or sound recording, the synchronization of the Work in timed-relation with a moving image ("synching") will be considered a Derivative Work for the purpose of this License.

c. **"Licensor"** means the individual or entity that offers the Work under the terms of this License.

d. **"Original Author"** means the individual or entity who created the Work.

e. **"Work"** means the copyrightable work of authorship offered under the terms of this License.

f. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright

Figure 8