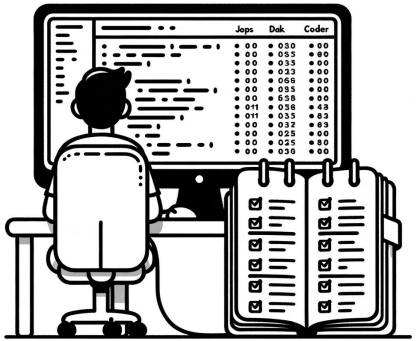
CodeGRITS: A Research Toolkit for Developer Behavior and Eye Tracking in IDE

Ningzhi Tang*, Junwen An*, Meng Chen, Aakash Bansal,
Yu Huang, Collin McMillan, Toby Jia-Jun Li

Track

Behavior



Gain

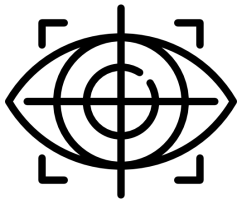
Insights

```
<Command __id="2" _type="MoveCaretCommand" caretOffset="142" docOffset="142" timestamp="3977"/>
<Command __id="3" _type="EclipseCommand" commandID="eventLogger.styledTextCommand.SELECT_LINE_DOWN"
timestamp="5598"/>
<DocumentChange __id="4" _type="Delete" docASTNodeCount="22" docActiveCodeLength="125" docExpression-
Count="10" docLength="151" endLine="9" length="39" offset="142" startLine="8" timestamp="7186">
  <text>
    <![CDATA[     System.out.println("Hello World!");

]]>
  </text>
</DocumentChange>
<Command __id="5" _type="EclipseCommand" commandID="org.eclipse.ui.edit.delete" timestamp="7202"/>
<Command __id="6" _type="EclipseCommand" commandID="org.eclipse.ui.file.save" timestamp="8099"/>
```

✔️ **What a programmer did?**

[1] Y. Yoon and B. A. Myers. 2011. Capturing and Analyzing Low-Level Events from the Code Editor. In *PLATEAU* '11.



Eye tracking

Understand

Cognitive processes

Visual attention, emotional arousal & cognitive workload

Infer

✔️ **<u>Why</u> they did it?**

Trustworthiness Perception
An Eye-tracking

ABSTRACT

An Empirical Study on the
Patterns of Eye Movement during
Summarization Tasks

Paige Rodeghero* and Collin McMillan*
*Department of Computer Science and Engineering

Towards Modeling Human Attention from Eye Movements
for Neural Source Code Summarization

AAKASH BANSAL, University of Notre Dame, USA
BONITA SHARIF, University of Nebraska - Lincoln, USA
COLLIN MCMILLAN, University of Notre Dame, USA

Neural source code summarization is the task of generating natural language descriptions of source code behavior using neural networks. A fundamental component of most neural models is an attention mechanism. The attention mechanism learns to connect features in source code to specific words to use when generating natural language descriptions. Humans also pay attention to some features in code more than others. This human attention reflects experience and high-level cognition well beyond the capability of any current neural model. In this paper, we use data from published eye-tracking experiments to create a model of this human attention. The model predicts which words in source code are the most important for code summarization. Next, we augment a baseline neural code summarization approach using our model of human attention. We observe an improvement in prediction performance of the augmented approach in line with other bio-inspired neural models.

CCS Concepts: • **Software and its engineering** → **Documentation**; **Maintaining software**; • **Computing methodologies** → **Natural language generation**; **Neural networks**.

Additional Key Words and Phrases: automatic documentation generation, source code summarization, bio-inspired models, eye tracking

ACM Reference Format:
Aakash Bansal, Bonita Sharif, and Collin McMillan. 2023. Towards Modeling Human Attention from Eye Movements for Neural Source Code Summarization. 1, 1 (May 2023), 19 pages. https://doi.org/10.1145/3591136

1 INTRODUCTION

Source code summarization is the task of writing natural language descriptions of source code. These descriptions are called "summaries" and are a key component of software documentation for programmers. A programmer may read a short summary like "takes a screenshot" to quickly understand what a section of code does, without resorting to reading the source code. Despite the usefulness of these summaries, programmers often neglect to write or update them. The result is that automatic source code summarization has long been an appetizing target in software engineering research. The scientific community has long sought to enable machines to understand code in the way people do, so that those machines can describe code like a person would.

A confluence of recent advances in both software engineering and machine learning research is bearing fruit, such that automated code summarization seems almost within reach. In particular, *neural* source code summarization has held the vanguard of the state of the art since around 2017. Neural code summarization refers to approaches based on neural networks, namely the encoder-decoder architecture [61]. The encoder-decoder architecture is borrowed from problem domains

Authors' addresses: Aakash Bansal, abansal1@nd.edu, University of Notre Dame, Department of Computer Science and Engineering, Notre Dame, Indiana, USA, 46656; Bonita Sharif, bsharif@unl.edu, University of Nebraska - Lincoln, School of Computing, Lincoln, Nebraska, USA, 68588; Collin McMillan, cmc@nd.edu, University of Notre Dame, Department of Computer Science and Engineering, Notre Dame, Indiana, USA, 46656.

This is a pre-print for paper to appear at PACM HCI 2023 Vol 7, in proceedings of ETRA '23: ACM Symposium on Eye Tracking Research Applications, May 30 - June 2, 2023, Tübingen, Germany
© 2023 All copyrights held by the authors.
XXXX-XXXX/2023/5-ART $15.00
https://doi.org/10.1145/3591136

[2] V. Skaramagkas et al . 2021. Review of Eye Tracking Metrics Involved in Emotional and Cognitive Processes. *IEEE Rev Biomed Eng* 16 (2021), 260–277.

Research Gap

Lacks support for JetBrains IDEs and multiple programming languages

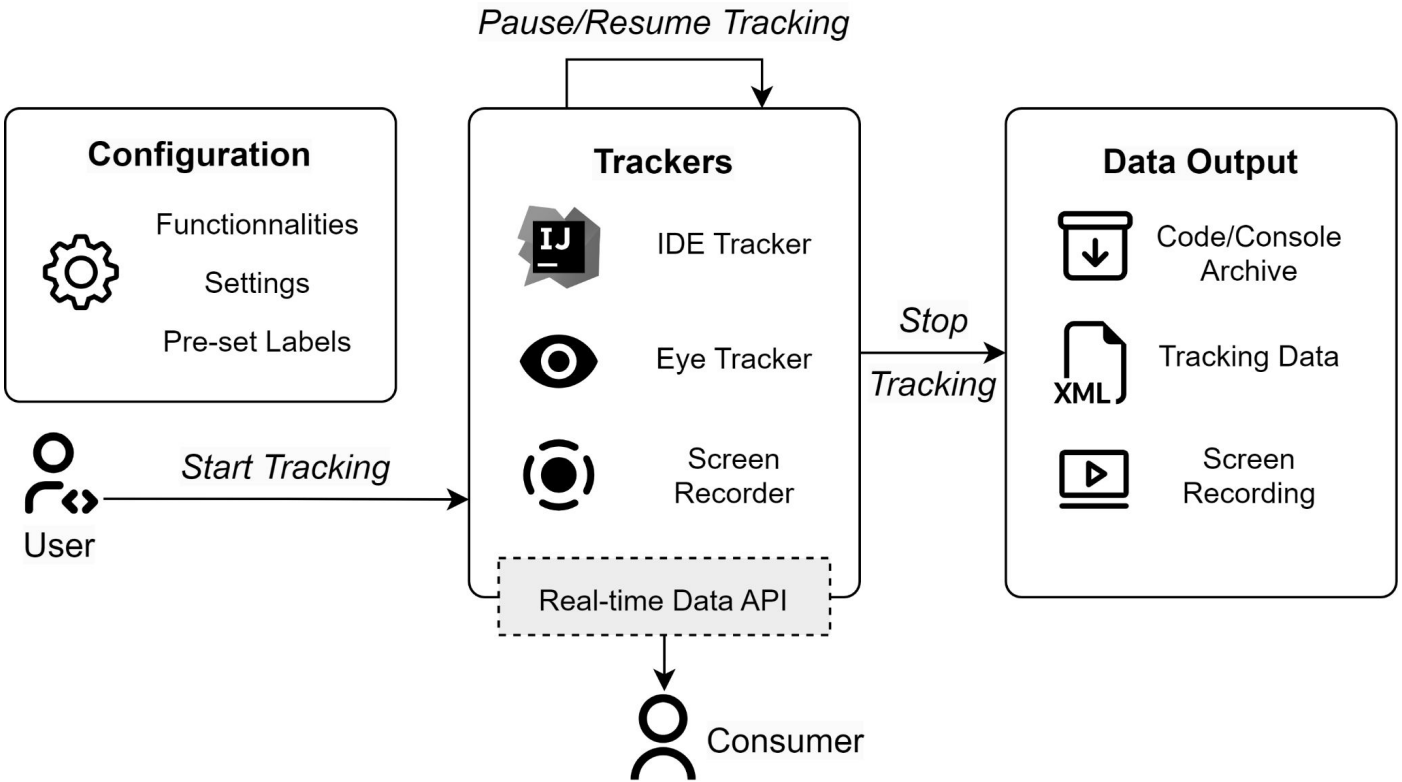
- **IntelliJ IDEA** – Java and Kotlin IDE
- **Clion** – C and C++ IDE
- **PyCharm** – Python IDE
- **PhpStorm** – PHP IDE

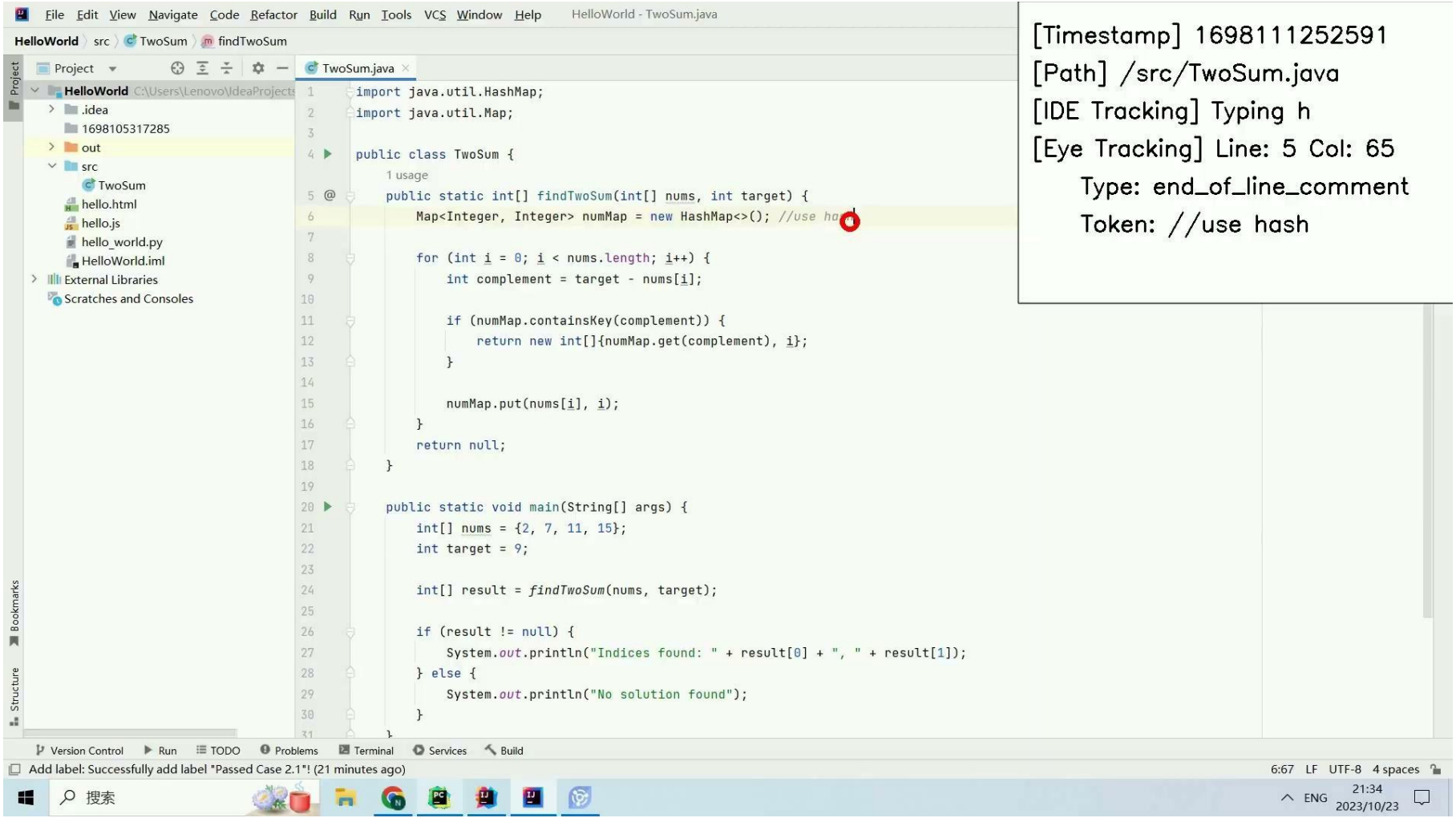
Cannot track multiple forms of behavioral data simultaneously

- **IDE Tracking**
- **Eye Tracking**
- **Screen Recording**

[3] D. T. Guarnera et al . 2018. ITrace: Eye Tracking Infrastructure for Development Environments. In *ETRA* '18.

CodeGRITS - Gaze Recording & IDE Tracking System






```java
import java.util.HashMap;
import java.util.Map;

public class TwoSum {
    1 usage
    public static int[] findTwoSum(int[] nums, int target) {
        Map<Integer, Integer> numMap = new HashMap<>(); //use ha

        for (int i = 0; i < nums.length; i++) {
            int complement = target - nums[i];

            if (numMap.containsKey(complement)) {
                return new int[]{numMap.get(complement), i};
            }

            numMap.put(nums[i], i);
        }
        return null;
    }

    public static void main(String[] args) {
        int[] nums = {2, 7, 11, 15};
        int target = 9;

        int[] result = findTwoSum(nums, target);

        if (result != null) {
            System.out.println("Indices found: " + result[0] + ", " + result[1]);
        } else {
            System.out.println("No solution found");
        }
    }
}
```

[Timestamp] 1698111252591
[Path] /src/TwoSum.java
[IDE Tracking] Typing h
[Eye Tracking] Line: 5 Col: 65
Type: end_of_line_comment
Token: //use hash

Data Output

```
[OUTPUT_DIR]
├── [START_TIMESTAMP]
|   ├── ide_tracking.xml
|   ├── eye_tracking.xml
|   ├── archives
|   |   ├── [ARCHIVE_TIMESTAMP_1].archive
|   |   ├── [ARCHIVE_TIMESTAMP_2].archive
|   |   ├── ...
|   ├── screen_recording
|   |   ├── video_clip_1.mp4
|   |   ├── video_clip_2.mp4
|   |   ├── ...
|   |   ├── frames.csv
```

```xml
<ide_tracking>
    <actions>
        <action id="SaveAll" path="/src/Main.java" timestamp="1696214490354"/>
        <action id="RunClass" path="/src/Main.java" timestamp="1696214496053"/>
        <action id="ToggleLineBreakpoint" path="/src/Main.java" timestamp="1696214500296"/>
        <action id="GotoDeclaration" path="/src/Main.java" timestamp="1696214513473"/>
        <action id="Debug" path="/src/Main.java" timestamp="1696216129173"/>
        <action id="NewClass" path="/src" timestamp="1696217116236"/>
        <action id="RenameElement" path="/src/ABC.java" timestamp="1696217122074"/>
    </actions>
    <typings>
        <typing character="S" column="8" line="3" path="/src/Main.java" timestamp="1696216429855"/>
        <typing character="y" column="9" line="3" path="/src/Main.java" timestamp="1696216430111"/>
    </typings>
    <files>
```

```xml
<eye_tracking>
    <gaze timestamp="1696224370377">
        <left_eye gaze_point_x="0.5338541666666666" gaze_point_y="0.17407407407407408" gaze_validity="1.0"
                  pupil_diameter="2.4835662841796875" pupil_validity="1.0"/>
        <right_eye gaze_point_x="0.5338541666666666" gaze_point_y="0.17407407407407408" gaze_validity="1.0"
                   pupil_diameter="2.7188568115234375" pupil_validity="1.0"/>
        <location column="25" line="2" path="/src/Main.java" x="820" y="150"/>
        <ast_structure token="println" type="IDENTIFIER">
            <level end="2:26" start="2:19" tag="PsiIdentifier:println"/>
            <level end="2:26" start="2:8" tag="PsiReferenceExpression:System.out.println"/>
            <level end="2:42" start="2:8" tag="PsiMethodCallExpression:System.out.println(&amp;quot;Hello world!&amp;quot;)"/>
            <level end="2:43" start="2:8" tag="PsiExpressionStatement"/>
            <level end="3:5" start="1:43" tag="PsiCodeBlock"/>
            <level end="3:5" start="1:4" tag="PsiMethod:main"/>
            <level end="4:1" start="0:0" tag="PsiClass:Main"/>
        </ast_structure>
    </gaze>
</eye_tracking>
```

Configuration

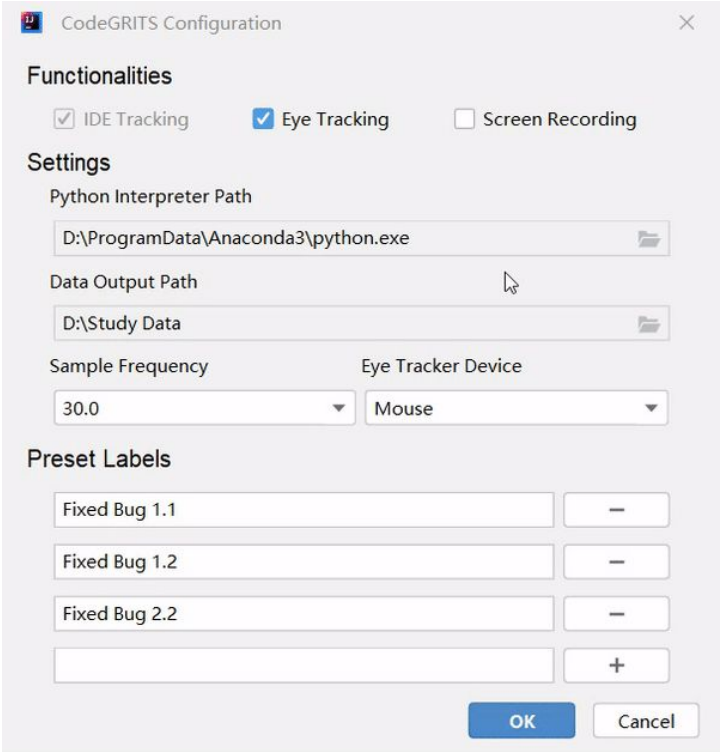
Functionalities

Select the trackers wanted to use.

Settings

Python Interpreter Path
Data Output Path
Eye Tracking Device
Sample Frequency

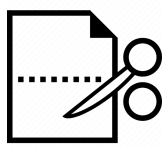
Preset Labels



IDE Tracker

Run

Debugger



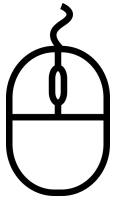
Clipboard

File

Timestamp

Typing

Caret

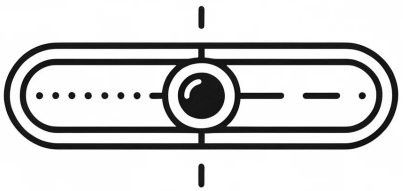


Mouse

Selection

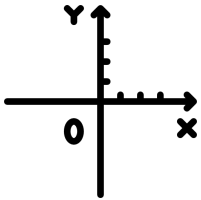
Location

Eye Tracker



Eye-tracking Device

Connecting



Raw Gaze Data
Coordinates & Pupil Size

Mapping

Location
Path, Line & Column

Eye Tracker

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Bottom-up Traversal of Abstract Syntax Tree (AST)

```xml
<ast_structure token="println" type="IDENTIFIER">
    <level end="2:26" start="2:19" tag="PsiIdentifier:println"/>
    <level end="2:26" start="2:8" tag="PsiReferenceExpression:System.out.println"/>
    <level end="2:42" start="2:8" tag="PsiMethodCallExpression:System.out.println(&quot;Hello world!&quot;)"/>
    <level end="2:43" start="2:8" tag="PsiExpressionStatement"/>
    <level end="3:5" start="1:43" tag="PsiCodeBlock"/>
    <level end="3:5" start="1:4" tag="PsiMethod:main"/>
    <level end="4:1" start="0:0" tag="PsiClass:Main"/>
</ast_structure>
```

AST Traversal in IDE-Supported Languages

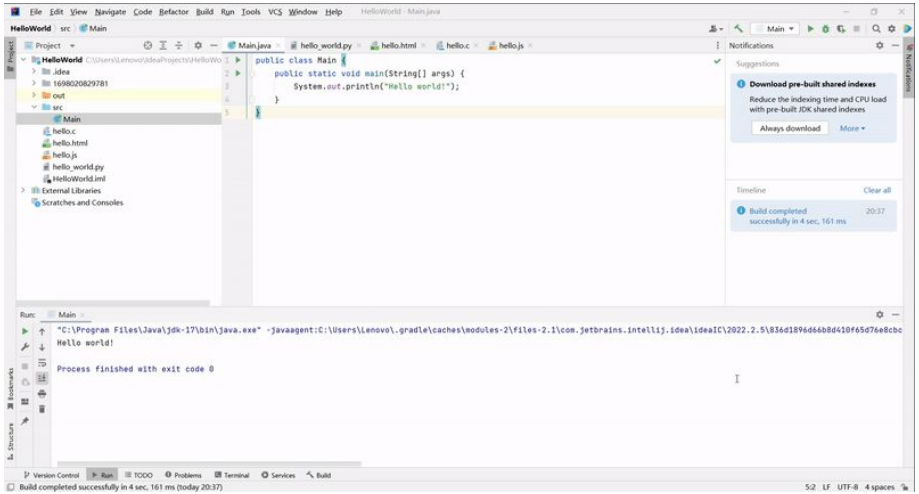
```
<location column="7" line="1" path="/hello_world.py" x="519" y="119"/>
<ast_structure token="print" type="Py:IDENTIFIER">
    <level end="1:9" start="1:4" tag="PsiElement(Py:IDENTIFIER)"/>
    <level end="1:9" start="1:4" tag="PyReferenceExpression: print"/>
    <level end="1:25" start="1:4" tag="PyCallExpression: print"/>
    <level end="1:25" start="1:4" tag="PyExpressionStatement"/>
    <level end="1:25" start="1:4" tag="PyStatementList"/>
    <level end="1:25" start="0:0" tag="PyFunction('hello_world')"/>
</ast_structure>
```

```
<location column="6" line="4" path="/hello.c" x="473" y="215"/>
<ast_structure token="return" type="OCKeyword:return">
    <level end="4:10" start="4:4" tag="PsiElement(OCKeyword:return)"/>
    <level end="4:13" start="4:4" tag="OCReturnStatement"/>
    <level end="5:1" start="2:11" tag="OCLazyBlockStatement"/>
    <level end="5:1" start="2:0" tag="OCFunctionDefinition(main)"/>
</ast_structure>
```


```
<location column="6" line="0" path="/hello.js" x="470" y="98"/>
<ast_structure token="&amp;quot;Hello, World!&amp;quot;" type="JS:STRING_LITERAL">
    <level end="0:21" start="0:6" tag="PsiElement(JS:STRING_LITERAL)"/>
    <level end="0:21" start="0:6" tag="JSLiteralExpression"/>
    <level end="0:22" start="0:5" tag="JSArgumentList"/>
    <level end="0:22" start="0:0" tag="JSCallExpression"/>
    <level end="0:23" start="0:0" tag="JSExpressionStatement"/>
</ast_structure>
```


```
<location column="14" line="4" path="/hello.html" x="585" y="176"/>
<ast_structure token="Hello" type="XML_DATA_CHARACTERS">
    <level end="4:16" start="4:11" tag="XmlToken:XML_DATA_CHARACTERS"/>
    <level end="4:22" start="4:11" tag="HtmlRawText"/>
    <level end="4:30" start="4:4" tag="HtmlTag:title"/>
    <level end="5:7" start="2:0" tag="HtmlTag:head"/>
    <level end="9:7" start="1:0" tag="HtmlTag:html"/>
    <level end="9:7" start="0:0" tag="PsiElement(HTML_DOCUMENT)"/>
</ast_structure>
```

Screen Recorder

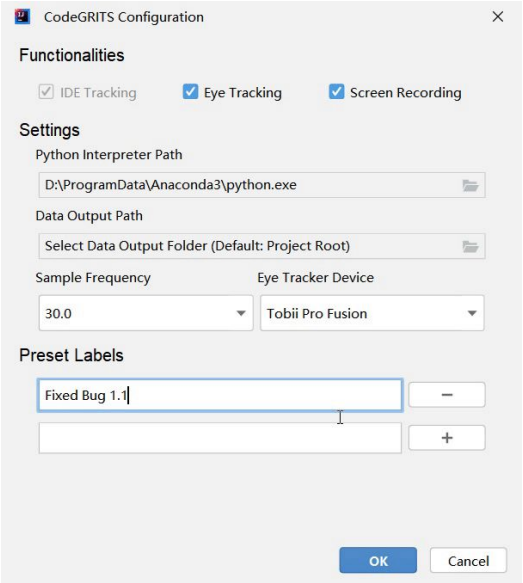


Video

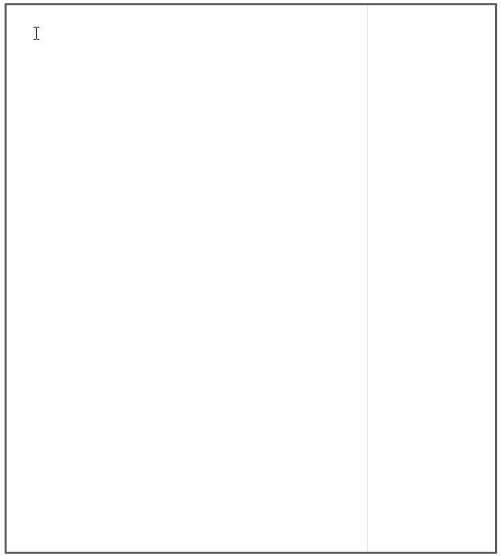
+

Timestamp of Each Frame

Adding Customizable Labels



Pre-set Labels



Add Label

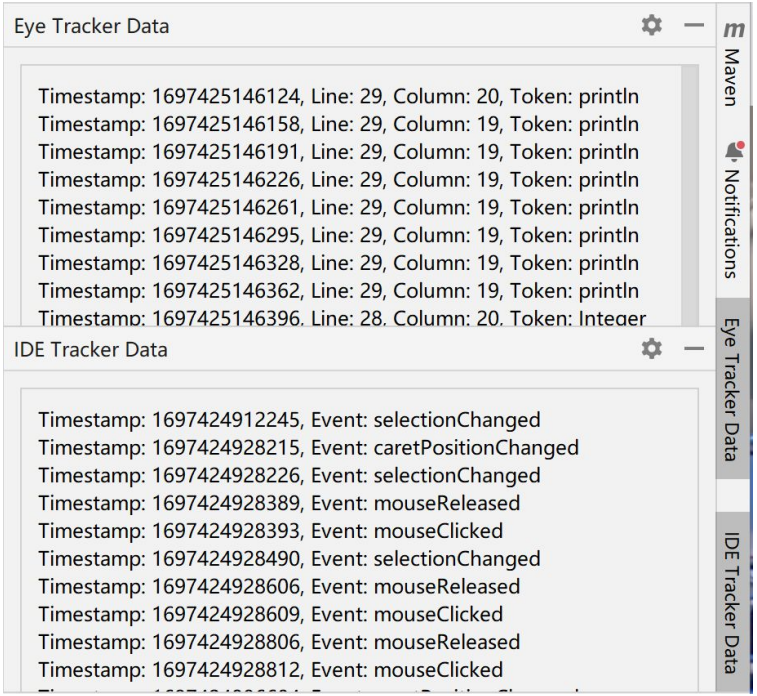
```
<action id="CodeVision.AddLabelAction.[Fixed Bug 1.2]"
        path="/src/Main.java" timestamp="1698033081816"/>
```

Data Output

Real-time Data API

```java
import trackers.EyeTracker;

public class EyeDataOutput {
    public void outputData(Project project) throws
            ParserConfigurationException,
            IOException {
        EyeTracker eyeTracker = new EyeTracker(
                pythonInterpreter: "D:\\Python\\python.exe",
                sampleFrequency: 30.0
        );
        EyeTracker.setIsRealTimeDataTransmitting(true);
        eyeTracker.setEyeTrackerDataHandler(element -> {
            System.out.println("element: " + element);
        });
        eyeTracker.startTracking(project);
    }
```



Use Cases

Understanding Developer Behavior and Cognition

Comprehension

Reviewing

Debugging

Quantitative Analysis

Findings

Context-aware Programming Support

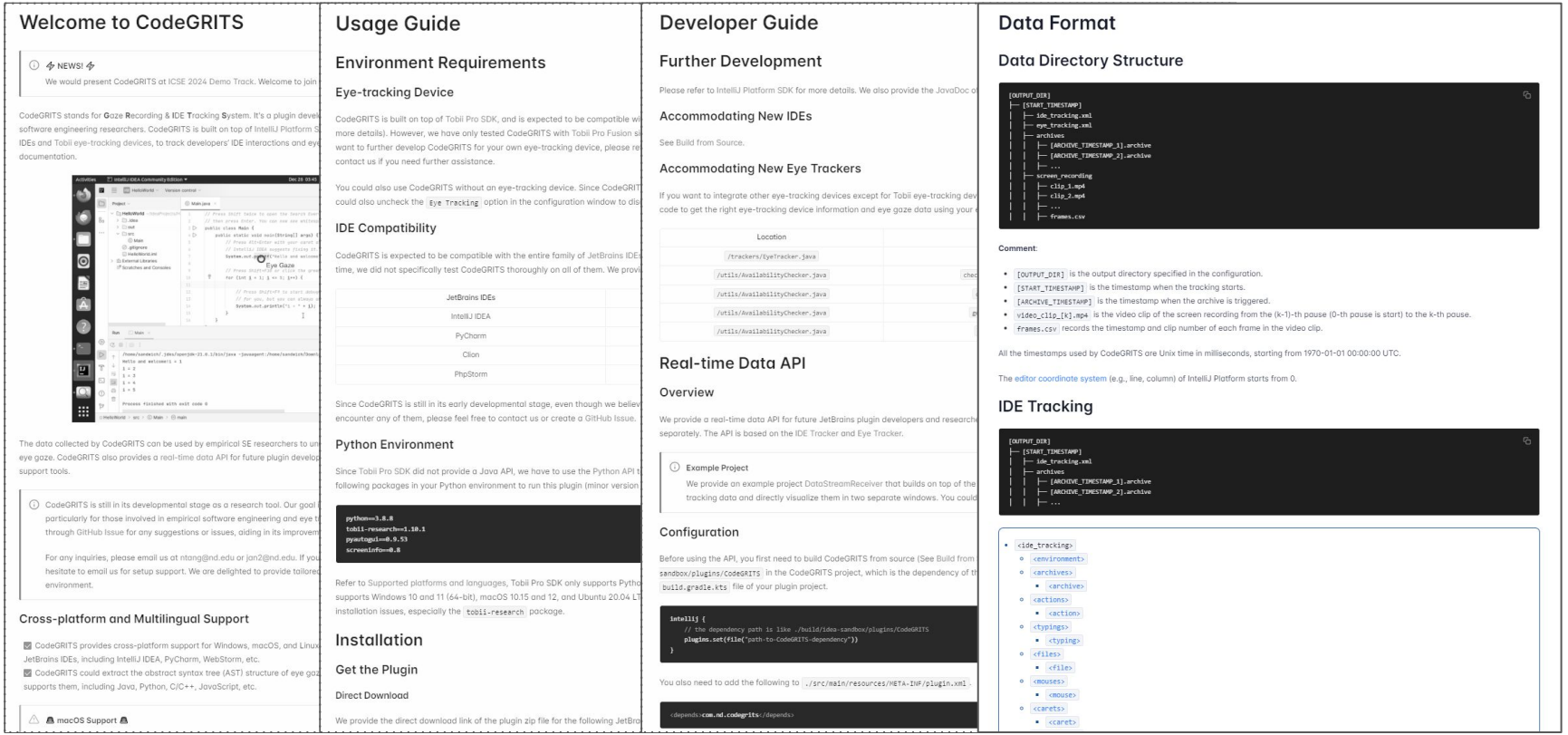
Behavior Pattern

Visual Attention

Context-aware Support

Programming Tools

Website



Welcome to CodeGRITS

NEWS!

We would present CodeGRITS at ICSE 2024 Demo Track. Welcome to join

CodeGRITS stands for **G**aze **R**ecording & IDE **T**racking **S**ystem. It's a plugin develo
software engineering researchers. CodeGRITS is built on top of IntelliJ Platform S
IDEs and Tobii eye-tracking devices, to track developers' IDE interactions and eye
documentation.

The data collected by CodeGRITS can be used by empirical SE researchers to un
eye gaze. CodeGRITS also provides a real-time data API for future plugin develop
support tools.

CodeGRITS is still in its developmental stage as a research tool. Our goal
particularly for those involved in empirical software engineering and eye t
through GitHub Issue for any suggestions or issues, aiding in its improvem

For any inquiries, please email us at ntang@nd.edu or jan2@nd.edu. If you
hesitate to email us for setup support. We are delighted to provide tailored
environment.

Cross-platform and Multilingual Support

CodeGRITS provides cross-platform support for Windows, macOS, and Linux
JetBrains IDEs, including IntelliJ IDEA, PyCharm, WebStorm, etc.
CodeGRITS could extract the obstract syntax tree (AST) structure of eye gaz
supports them, including Java, Python, C/C++, JavaScript, etc.

macOS Support

Usage Guide

Environment Requirements

Eye-tracking Device

CodeGRITS is built on top of Tobii Pro SDK, and is expected to be compatible wi
more details). However, we have only tested CodeGRITS with Tobii Pro Fusion si
want to further develop CodeGRITS for your own eye-tracking device, please re
contact us if you need further assistance.

You could also use CodeGRITS without an eye-tracking device. Since CodeGRIT
could also uncheck the `Eye Tracking` option in the configuration window to dis

IDE Compatibility

CodeGRITS is expected to be compatible with the entire family of JetBrains IDEs
time, we did not specifically test CodeGRITS thoroughly on all of them. We provi

JetBrains IDEs
IntelliJ IDEA
PyCharm
Clion
PhpStorm

Since CodeGRITS is still in its early developmental stage, even though we believ
encounter any of them, please feel free to contact us or create a GitHub Issue.

Python Environment

Since Tobii Pro SDK did not provide a Java API, we have to use the Python API t
following packages in your Python environment to run this plugin (minor version

```
python==3.8.8
tobii-research==1.10.1
pyautogui==0.9.53
screeninfo==0.8
```

Refer to Supported platforms and languages, Tobii Pro SDK only supports Pytho
supports Windows 10 and 11 (64-bit), macOS 10.15 and 12, and Ubuntu 20.04 LT
installation issues, especially the `tobii-research` package.

Installation

Get the Plugin

Direct Download

We provide the direct download link of the plugin zip file for the following JetBra

Developer Guide

Further Development

Please refer to IntelliJ Platform SDK for more details. We also provide the JavaDoc o

Accommodating New IDEs

See Build from Source.

Accommodating New Eye Trackers

If you want to integrate other eye-tracking devices except for Tobii eye-tracking dev
code to get the right eye-tracking device information and eye gaze data using your

Location
`/trackers/EyeTracker.java`
`/utils/AvailabilityChecker.java`
`/utils/AvailabilityChecker.java`
`/utils/AvailabilityChecker.java`
`/utils/AvailabilityChecker.java`

Real-time Data API

Overview

We provide a real-time data API for future JetBrains plugin developers and researche
separately. The API is based on the IDE Tracker and Eye Tracker.

Example Project

We provide an example project DataStreamReceiver that builds on top of the
tracking data and directly visualize them in two separate windows. You could

Configuration

Before using the API, you first need to build CodeGRITS from source (See Build from
`sandbox/plugins/CodeGRITS` in the CodeGRITS project, which is the dependency of th
`build.gradle.kts` file of your plugin project.

```
intellij {
    // the dependency path is like ./build/idea-sandbox/plugins/CodeGRITS
    plugins.set(file("path-to-CodeGRITS-dependency"))
}
```

You also need to add the following to `./src/main/resources/META-INF/plugin.xml`.

```
<depends>com.nd.codegrits</depends>
```

Data Format

Data Directory Structure

```
[OUTPUT_DIR]
├── [START_TIMESTAMP]
│   ├── ide_tracking.xml
│   ├── eye_tracking.xml
│   ├── archives
│   │   ├── [ARCHIVE_TIMESTAMP_1].archive
│   │   ├── [ARCHIVE_TIMESTAMP_2].archive
│   │   ├── ...
│   ├── screen_recording
│   │   ├── clip_1.mp4
│   │   ├── clip_2.mp4
│   │   ├── ...
│   │   ├── frames.csv
```

Comment:

- `[OUTPUT_DIR]` is the output directory specified in the configuration.
- `[START_TIMESTAMP]` is the timestamp when the tracking starts.
- `[ARCHIVE_TIMESTAMP]` is the timestamp when the archive is triggered.
- `video_clip_[k].mp4` is the video clip of the screen recording from the (k-1)-th pause (0-th pause is start) to the k-th pause.
- `frames.csv` records the timestamp and clip number of each frame in the video clip.

All the timestamps used by CodeGRITS are Unix time in milliseconds, starting from 1970-01-01 00:00:00 UTC.

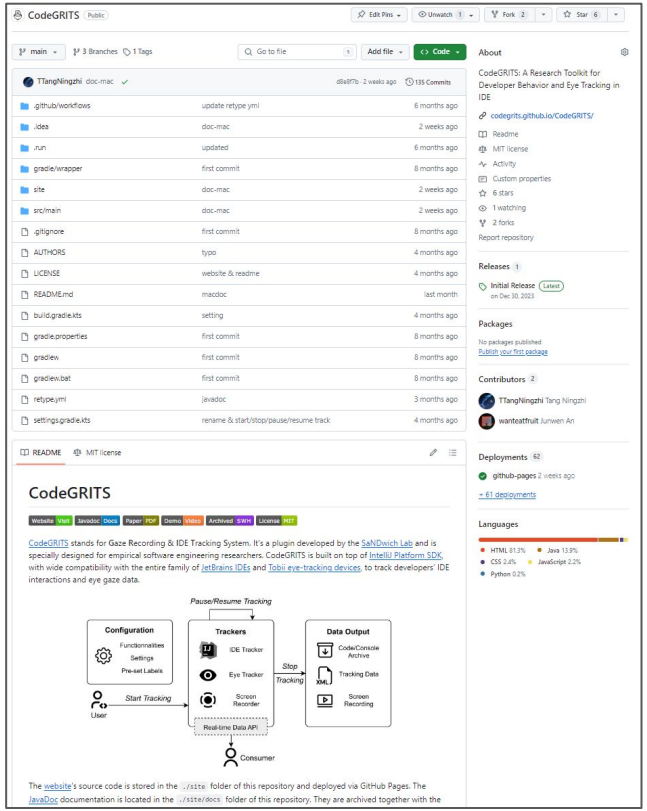
The editor coordinate system (e.g., line, column) of IntelliJ Platform starts from 0.

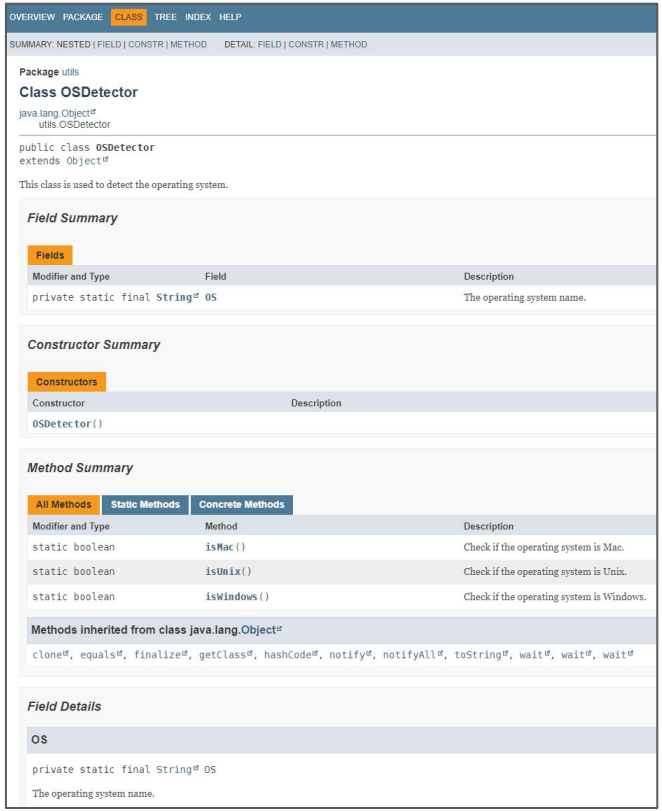
IDE Tracking

```
[OUTPUT_DIR]
├── [START_TIMESTAMP]
│   ├── ide_tracking.xml
│   ├── archives
│   │   ├── [ARCHIVE_TIMESTAMP_1].archive
│   │   ├── [ARCHIVE_TIMESTAMP_2].archive
│   │   ├── ...
```

- <ide_tracking>
  - <environment>
  - <archives>
    - <archive>
  - <actions>
    - <action>
  - <typings>
    - <typing>
  - <files>
    - <file>
  - <mouses>
    - <mouse>
  - <carets>
    - <caret>

Availability



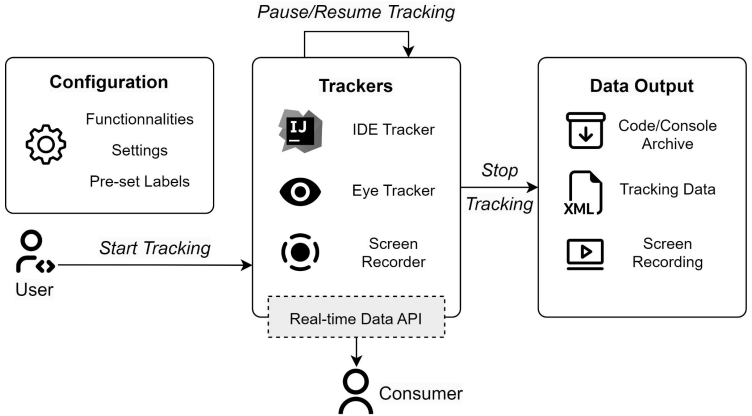


CodeGRITS: A Research Toolkit for Developer Behavior and Eye Tracking in IDE

Ningzhi Tang*, Junwen An*, Meng Chen, Aakash Bansal, Yu Huang, Collin McMillan, Toby Jia-Jun Li
{ntang, jan2, mchen24, abansal1, cmc, toby.j.li}@nd.edu, yu.huang@vanderbilt.edu

Ningzhi Tang

nztang.com
ntang@nd.edu
@TangNingzhi
ningzhi_tang



1. **Simultaneously tracks** developers' IDE interactions and eye-tracking data.
2. Provides **extra features** to fulfill the needs of empirical SE researchers.
3. Provides **wide compatibility** with different JetBrains IDEs and programming languages.

codegrits.github.io/CodeGRITS