

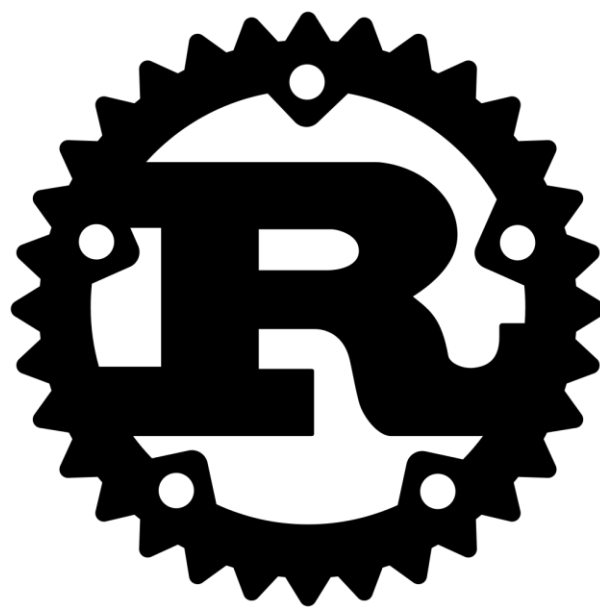


Developing Mozilla’s Servo Using CO-FOSS Principles

Team Seawolves-Again-Redux

Shaina Mainar, David Spry, Chandler Long, Cory Goodwin

Department of Computer Science, College of Charleston, Charleston, SC



Abstract

Mozilla is a company focused on open-source development of many popular software including Firefox and Thunderbird. This project is focused on the development of Servo, Mozilla’s Rust-based web browser, to implement a DelayNode node in the web browser for the purpose of developing the browser’s audio capabilities. This project was completed under the instruction of Dr. Bowring.

Goals


- Learn Rust, a Mozilla-developed language
- Develop the code for the audio node
- Use the included example code to test the functionality of the node
- Push code to GitHub after approval

Motivation

- Computer Science students need to be able to prepare for industry by utilizing tools and enhancing skillsets necessary to succeed.
- Hands-on experience with the Software Engineering Life Cycle.

Conclusion

In the end, we were able to develop the DelayNode, but due to the underestimated challenge of working in a new code base and a lack of clarity with regards to the guidelines for the implementation, we couldn’t get every test running smoothly, so we decided to push what we had in hopes that it can still be picked up by another team of developers and completed. Overall, we learned a lot about developing in a CO-FOSS environment and developed some connections and skills along the way.

 servo / servo

<> Code

Issues 3,110

Pull requests 62

Actions

Projects 21

Implement DelayNode node #22369

Open

ferjm opened this issue on Dec 5, 2018 · 4 comments

Methodology

Testing:

- Firstly, we built Servo and Servo Media as instructed on their GitHub page.
- Next, we began to learn Rust, as we were all new to the programming language.
- Then, we studied audio functionality in order to properly implement the DelayNode.
- Finally, we started developing the implementation.
- Overall, the biggest challenge was testing our implementation. We were able to run some tests successfully, but not all.

Code Snippet

```
//use node::{AudioNode, AudioParam, AudioNodeEngine};
use node::{AudioNodeEngine, AudioNodeMessage, BlockInfo};
use block::{Block, Chunk, Tick};
use param::{Param, ParamType};
use node::{AudioNodeType, ChannelInfo};

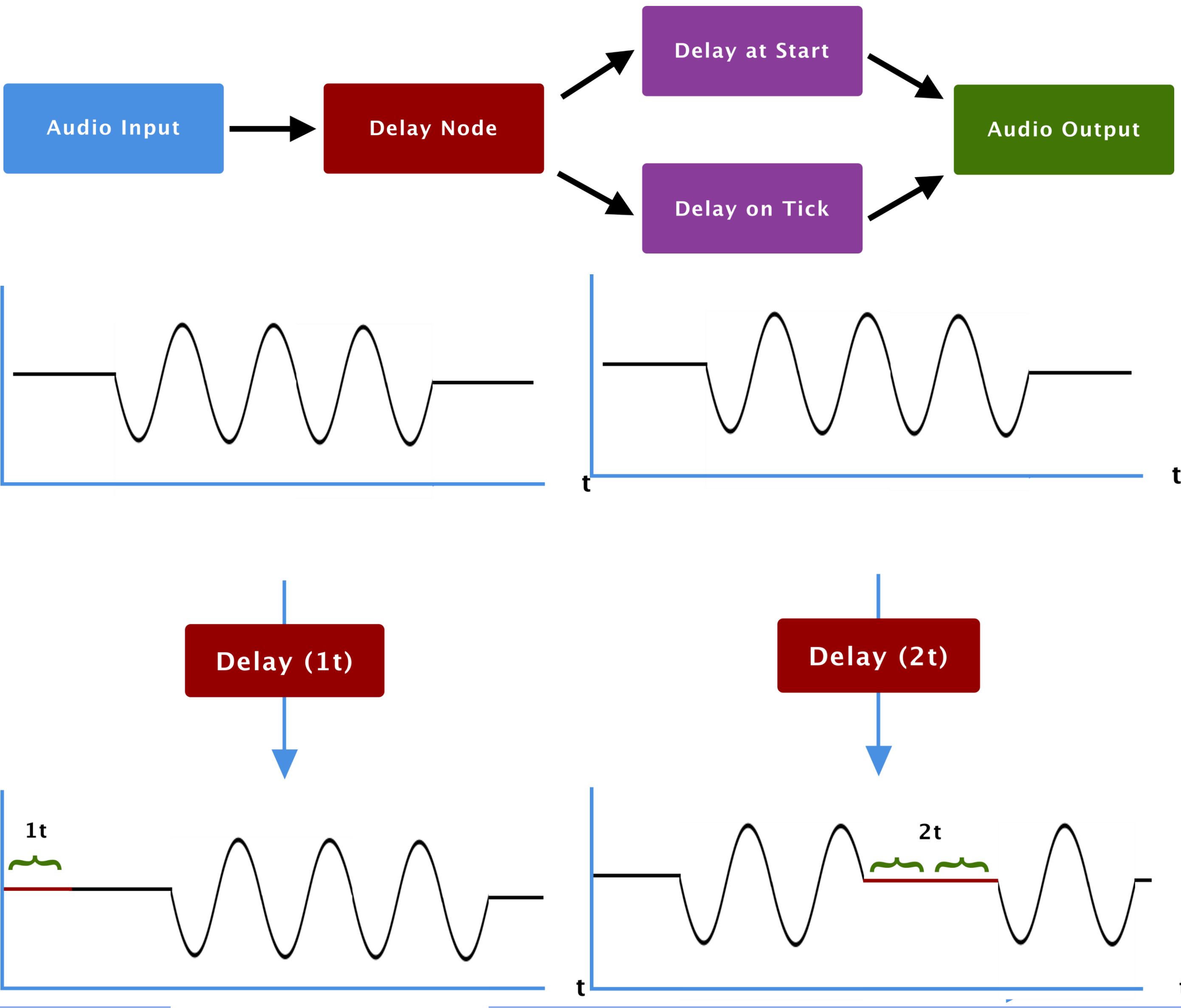
/* pub struct DelayNode{
    pub c: BaseAudioContext,
    pub option: DelayOptions
} */

pub struct DelayOptions{
    pub maxDelayTime: f32,
    pub delayTime: f32,
}

impl Default for DelayOptions{
    fn default() -> Self{
        DelayOptions{
            maxDelayTime: 1.0,
            delayTime: 0.0,
        }
    }
}

//Constructor (line 173 DelayNode.cpp)
//Note: find implementation in servo/servo
#[derive(AudioNodeCommon)]
pub(crate) struct DelayNode{
    delay_type: DelayType,
    delayTime: Option<Tick>,
    channel_info: ChannelInfo,
}
```

What is a DelayNode?



Retrospect

We weren’t sure where to start at first. Our task was simply to find an open-source software and contribute to it, so we initially chose Mozilla’s Firefox as a place to start. We started by looking at fixing bugs, but our experience with their bug-reporting website wasn’t looking promising. We followed up on the potential project of repairing their developer VM, but in following up with as many people as we could for more information, we were told that the VM project wasn’t a good fit for us. We were then assigned to help with Servo with some mentorship help. The process of getting to this point is quite simplified here, but it took a couple of months before we started looking at Servo. After some issues getting started and troubleshooting amongst ourselves, we began development on the node and learned a lot about audio development, as well as development in Rust, with some guidance. We’re happy to have gained the experience and hope that our developments will be of use to the Mozilla team.