# Radium: A Music Editor Inspired by the Music Tracker

Kjetil Matheussen

Norwegian Center for Technology in Music and the Arts (NOTAM)

May 3, 2014

## Introduction to Radium

- A music editor.
  - Made for composing music.
  - Interface inspired by the tracker interface.

  1. First stable version released in 2000, for AmigaOS.
  2. Pre-alpha version for Linux available in 2001
  3. First usable version for Linux available in 2005. Called "E-Radium".
  4. First non-alpha **native** Linux version released in 2012.
- The two major dependencies for Radium are Jack and Qt.

## Introduction to Radium

- A music editor.
  - Made for composing music.
  - Interface inspired by the tracker interface.

  1. First stable version released in 2000, for AmigaOS.
  2. Pre-alpha version for Linux available in 2001
  3. First usable version for Linux available in 2005. Called “E-Radium”.
  4. First non-alpha **native** Linux version released in 2012.
- The two major dependencies for Radium are Jack and Qt.

## Introduction to Radium

- A music editor.
  - Made for composing music.
  - Interface inspired by the tracker interface.

  1. First stable version released in 2000, for AmigaOS.
  2. Pre-alpha version for Linux available in 2001
  3. First usable version for Linux available in 2005. Called “E-Radium”.
  4. First non-alpha **native** Linux version released in 2012.

- The two major dependencies for Radium are Jack and Qt.

## Introduction to Radium

- A music editor.
  - Made for composing music.
  - Interface inspired by the tracker interface.

  1. First stable version released in 2000, for AmigaOS.
  2. Pre-alpha version for Linux available in 2001
  3. First usable version for Linux available in 2005. Called “E-Radium”.
  4. First non-alpha **native** Linux version released in 2012.
- The two major dependencies for Radium are Jack and Qt.

## Introduction to Radium

- A music editor.
  - Made for composing music.
  - Interface inspired by the tracker interface.

  1. First stable version released in 2000, for AmigaOS.
  2. Pre-alpha version for Linux available in 2001
  3. First usable version for Linux available in 2005. Called "E-Radium".
  4. First non-alpha **native** Linux version released in 2012.
- The two major dependencies for Radium are Jack and Qt.

## Introduction to Radium

- A music editor.
  - Made for composing music.
  - Interface inspired by the tracker interface.

  1. First stable version released in 2000, for AmigaOS.
  2. Pre-alpha version for Linux available in 2001
  3. First usable version for Linux available in 2005. Called "E-Radium".
  4. First non-alpha **native** Linux version released in 2012.
- The two major dependencies for Radium are Jack and Qt.

## Introduction to Radium

- A music editor.
  - Made for composing music.
  - Interface inspired by the tracker interface.

  1. First stable version released in 2000, for AmigaOS.
  2. Pre-alpha version for Linux available in 2001
  3. First usable version for Linux available in 2005. Called "E-Radium".
  4. First non-alpha **native** Linux version released in 2012.
- The two major dependencies for Radium are Jack and Qt.

## Introduction to Radium

- A music editor.
  - Made for composing music.
  - Interface inspired by the tracker interface.

  1. First stable version released in 2000, for AmigaOS.
  2. Pre-alpha version for Linux available in 2001
  3. First usable version for Linux available in 2005. Called "E-Radium".
  4. First non-alpha **native** Linux version released in 2012.
- The two major dependencies for Radium are Jack and Qt.

## Introduction to Radium

- A music editor.
  - Made for composing music.
  - Interface inspired by the tracker interface.

  1. First stable version released in 2000, for AmigaOS.
  2. Pre-alpha version for Linux available in 2001
  3. First usable version for Linux available in 2005. Called “E-Radium”.
  4. First non-alpha **native** Linux version released in 2012.
- The two major dependencies for Radium are Jack and Qt.

## Introduction to Radium

- A music editor.
  - Made for composing music.
  - Interface inspired by the tracker interface.

  1. First stable version released in 2000, for AmigaOS.
  2. Pre-alpha version for Linux available in 2001
  3. First usable version for Linux available in 2005. Called “E-Radium”.
  4. First non-alpha **native** Linux version released in 2012.
- The two major dependencies for Radium are Jack and Qt.

## Introduction to Music Trackers

- A type of music editor
- Editor is a two-dimentional table
  - The cells in this table only contains text.
  - tracks as columns
  - lines as rows (time)
- Time goes downwards
- Cursor always in a fixed position in the middle of the screen
- 80s and 90s on the Amiga and PC.
- First tracker called Ultimate Soundtracker, Karsten Obarski, 1987.

## Introduction to Music Trackers

- A type of music editor
- Editor is a two-dimentional table
  - The cells in this table only contains text.
  - tracks as columns
  - lines as rows (time)
- Time goes downwards
- Cursor always in a fixed position in the middle of the screen
- 80s and 90s on the Amiga and PC.
- First tracker called Ultimate Soundtracker, Karsten Obarski, 1987.

## Introduction to Music Trackers

- A type of music editor
- Editor is a two-dimentional table
  - The cells in this table only contains text.
  - tracks as columns
  - lines as rows (time)
- Time goes downwards
- Cursor always in a fixed position in the middle of the screen
- 80s and 90s on the Amiga and PC.
- First tracker called Ultimate Soundtracker, Karsten Obarski, 1987.

## Introduction to Music Trackers

- A type of music editor
- Editor is a two-dimentional table
  - The cells in this table only contains text.
  - tracks as columns
  - lines as rows (time)
- Time goes downwards
- Cursor always in a fixed position in the middle of the screen
- 80s and 90s on the Amiga and PC.
- First tracker called Ultimate Soundtracker, Karsten Obarski, 1987.

## Introduction to Music Trackers

- A type of music editor
- Editor is a two-dimentional table
  - The cells in this table only contains text.
  - tracks as columns
  - lines as rows (time)
- Time goes downwards
- Cursor always in a fixed position in the middle of the screen
- 80s and 90s on the Amiga and PC.
- First tracker called Ultimate Soundtracker, Karsten Obarski, 1987.

## Introduction to Music Trackers

- A type of music editor
- Editor is a two-dimentional table
  - The cells in this table only contains text.
  - tracks as columns
  - lines as rows (time)
- Time goes downwards
- Cursor always in a fixed position in the middle of the screen
- 80s and 90s on the Amiga and PC.
- First tracker called Ultimate Soundtracker, Karsten Obarski, 1987.

## Introduction to Music Trackers

- A type of music editor
- Editor is a two-dimentional table
  - The cells in this table only contains text.
  - tracks as columns
  - lines as rows (time)
- Time goes downwards
- Cursor always in a fixed position in the middle of the screen
- 80s and 90s on the Amiga and PC.
- First tracker called Ultimate Soundtracker, Karsten Obarski, 1987.

## Introduction to Music Trackers

- A type of music editor
- Editor is a two-dimentional table
  - The cells in this table only contains text.
  - tracks as columns
  - lines as rows (time)
- Time goes downwards
- Cursor always in a fixed position in the middle of the screen
- 80s and 90s on the Amiga and PC.
- First tracker called Ultimate Soundtracker, Karsten Obarski, 1987.

## Introduction to Music Trackers

- A type of music editor
- Editor is a two-dimentional table
  - The cells in this table only contains text.
  - tracks as columns
  - lines as rows (time)
- Time goes downwards
- Cursor always in a fixed position in the middle of the screen
- 80s and 90s on the Amiga and PC.
- First tracker called Ultimate Soundtracker, Karsten Obarski, 1987.

## Introduction to Music Trackers

- A type of music editor
- Editor is a two-dimentional table
  - The cells in this table only contains text.
  - tracks as columns
  - lines as rows (time)
- Time goes downwards
- Cursor always in a fixed position in the middle of the screen
- 80s and 90s on the Amiga and PC.
- First tracker called Ultimate Soundtracker, Karsten Obarski, 1987.

## How Radium is different from a music tracker

- Using graphical elements instead of text
- Allowing any number of events to be placed anywhere
  - A line in Radium is essentially just a graphical hint.
- Is Radium a tracker?

## How Radium is different from a music tracker

- Using graphical elements instead of text
- Allowing any number of events to be placed anywhere
  - A line in Radium is essentially just a graphical hint.
- Is Radium a tracker?

## How Radium is different from a music tracker

- Using graphical elements instead of text
- Allowing any number of events to be placed anywhere
    - A line in Radium is essentially just a graphical hint.
- Is Radium a tracker?

## How Radium is different from a music tracker

- Using graphical elements instead of text
- Allowing any number of events to be placed anywhere
    - A line in Radium is essentially just a graphical hint.
- Is Radium a tracker?

## How Radium is different from a music tracker

- Using graphical elements instead of text
- Allowing any number of events to be placed anywhere
    - A line in Radium is essentially just a graphical hint.
- Is Radium a tracker?

## Editor Features 1/2

- Time-varying volume changes
- Time-varying tempo changes
- Time-varying pitch changes
- Automation
- Micro-tonality
- Line splitting
- Zoom in out
- Undo/redo

## Editor Features 1/2

- Time-varying volume changes
- Time-varying tempo changes
- Time-varying pitch changes
- Automation
- Micro-tonality
- Line splitting
- Zoom in out
- Undo/redo

## Editor Features 1/2

- Time-varying volume changes
- Time-varying tempo changes
- Time-varying pitch changes
- Automation
- Micro-tonality
- Line splitting
- Zoom in out
- Undo/redo

## Editor Features 1/2

- Time-varying volume changes
- Time-varying tempo changes
- Time-varying pitch changes
- Automation
- Micro-tonality
- Line splitting
- Zoom in out
- Undo/redo

## Editor Features 1/2

- Time-varying volume changes
- Time-varying tempo changes
- Time-varying pitch changes
- Automation
- Micro-tonality
- Line splitting
- Zoom in out
- Undo/redo

## Editor Features 1/2

- Time-varying volume changes
- Time-varying tempo changes
- Time-varying pitch changes
- Automation
- Micro-tonality
- Line splitting
- Zoom in out
- Undo/redo

## Editor Features 1/2

- Time-varying volume changes
- Time-varying tempo changes
- Time-varying pitch changes
- Automation
- Micro-tonality
- Line splitting
- Zoom in out
- Undo/redo

## Editor Features 1/2

- Time-varying volume changes
- Time-varying tempo changes
- Time-varying pitch changes
- Automation
- Micro-tonality
- Line splitting
- Zoom in out
- Undo/redo

## Editor Features 1/2

- Time-varying volume changes
- Time-varying tempo changes
- Time-varying pitch changes
- Automation
- Micro-tonality
- Line splitting
- Zoom in out
- Undo/redo

## Editor features 2/2

- Glissando
- Invert
- Backwards
- Scripting
- Color configuration

## Editor features 2/2

- Glissando
- Invert
- Backwards
- Scripting
- Color configuration

## Editor features 2/2

- Glissando
- Invert
- Backwards
- Scripting
- Color configuration

## Editor features 2/2

- Glissando
- Invert
- Backwards
- Scripting
- Color configuration

## Editor features 2/2

- Glissando
- Invert
- Backwards
- Scripting
- Color configuration

## Editor features 2/2

- Glissando
- Invert
- Backwards
- Scripting
- Color configuration

## The modular mixer

- Connect audio
- Connect events

## The modular mixer

- Connect audio
- Connect events

## The modular mixer

- Connect audio
- Connect events

## The compressor interface

## STK Instruments

- 20 STK instruments doing physical modeling (Cook/Scavone).
- Implementation by Romain Michon in the Faust language.
- Michon's instruments have been slightly modified to be used as instruments in Radium.
    - Any Faust instrument that provides "gate", "freq" and "gain" controls can easily be used as polyphonic instruments in Radium.

## STK Instruments

- 20 STK instruments doing physical modeling (Cook/Scavone).
- Implementation by Romain Michon in the Faust language.
- Michon's instruments have been slightly modified to be used as instruments in Radium.
  - Any Faust instrument that provides "gate", "freq" and "gain" controls can easily be used as polyphonic instruments in Radium.

## STK Instruments

- 20 STK instruments doing physical modeling (Cook/Scavone).
- Implementation by Romain Michon in the Faust language.
- Michon's instruments have been slightly modified to be used as instruments in Radium.
  - Any Faust instrument that provides "gate", "freq" and "gain" controls can easily be used as polyphonic instruments in Radium.

## STK Instruments

- 20 STK instruments doing physical modeling (Cook/Scavone).
- Implementation by Romain Michon in the Faust language.
- Michon's instruments have been slightly modified to be used as instruments in Radium.
  - Any Faust instrument that provides "gate", "freq" and "gain" controls can easily be used as polyphonic instruments in Radium.

## STK Instruments

- 20 STK instruments doing physical modeling (Cook/Scavone).
- Implementation by Romain Michon in the Faust language.
- Michon's instruments have been slightly modified to be used as instruments in Radium.
  - Any Faust instrument that provides "gate", "freq" and "gain" controls can easily be used as polyphonic instruments in Radium.

## Common Music Notation

CMN: https://ccrma.stanford.edu/software/cmn/cmn/cmn.html
Common lisp package for generating western style scores.
CMN has support for Radium songs.

## Embedding Pure Data

- Uses the wrapper code in libpd to embed Pd.
- Running several Pd instances simultaneously are achieved by loading each libpd instance with the RTLD_LOCAL flag.

  (Available as a separate library called *libpds*: `https://github.com/kmatheussen/libpd`)
- Features:
  1. Process audio
  2. Controllers: Int, Float and Bool
  3. Process Note events (frame accurately)
  4. Process Velocity events (frame accurately)
  5. Process Pitch events (frame accurately)

## Embedding Pure Data

- Uses the wrapper code in libpd to embed Pd.
- Running several Pd instances simultaneously are achieved by loading each libpd instance with the RTLD_LOCAL flag.

  (Available as a separate library called *libpds*: `https://github.com/kmatheussen/libpd`)
- Features:
  1. Process audio
  2. Controllers: Int, Float and Bool
  3. Process Note events (frame accurately)
  4. Process Velocity events (frame accurately)
  5. Process Pitch events (frame accurately)

## Embedding Pure Data

- Uses the wrapper code in libpd to embed Pd.
- Running several Pd instances simultaneously are achieved by loading each libpd instance with the RTLD_LOCAL flag.

  (Available as a separate library called *libpds*: `https://github.com/kmatheussen/libpd`)
- Features:
  1. Process audio
  2. Controllers: Int, Float and Bool
  3. Process Note events (frame accurately)
  4. Process Velocity events (frame accurately)
  5. Process Pitch events (frame accurately)

## Embedding Pure Data

- Uses the wrapper code in libpd to embed Pd.
- Running several Pd instances simultaneously are achieved by loading each libpd instance with the RTLD_LOCAL flag.

  (Available as a separate library called *libpds*: `https://github.com/kmatheussen/libpd`)
- Features:
  1. Process audio
  2. Controllers: Int, Float and Bool
  3. Process Note events (frame accurately)
  4. Process Velocity events (frame accurately)
  5. Process Pitch events (frame accurately)

## Embedding Pure Data

- Uses the wrapper code in libpd to embed Pd.
- Running several Pd instances simultaneously are achieved by loading each libpd instance with the RTLD_LOCAL flag.

  (Available as a separate library called *libpds*: `https://github.com/kmatheussen/libpd`)
- Features:
  1. Process audio
  2. Controllers: Int, Float and Bool
  3. Process Note events (frame accurately)
  4. Process Velocity events (frame accurately)
  5. Process Pitch events (frame accurately)

## Embedding Pure Data

- Uses the wrapper code in libpd to embed Pd.
- Running several Pd instances simultaneously are achieved by loading each libpd instance with the RTLD_LOCAL flag.

  (Available as a separate library called *libpds*: `https://github.com/kmatheussen/libpd`)
- Features:
  1. Process audio
  2. Controllers: Int, Float and Bool
  3. Process Note events (frame accurately)
  4. Process Velocity events (frame accurately)
  5. Process Pitch events (frame accurately)

## Embedding Pure Data

- Uses the wrapper code in libpd to embed Pd.
- Running several Pd instances simultaneously are achieved by loading each libpd instance with the RTLD_LOCAL flag.

  (Available as a separate library called *libpds*: `https://github.com/kmatheussen/libpd`)
- Features:
  1. Process audio
  2. Controllers: Int, Float and Bool
  3. Process Note events (frame accurately)
  4. Process Velocity events (frame accurately)
  5. Process Pitch events (frame accurately)

## Embedding Pure Data

- Uses the wrapper code in libpd to embed Pd.
- Running several Pd instances simultaneously are achieved by loading each libpd instance with the RTLD_LOCAL flag.

  (Available as a separate library called *libpds*: `https://github.com/kmatheussen/libpd`)
- Features:
  1. Process audio
  2. Controllers: Int, Float and Bool
  3. Process Note events (frame accurately)
  4. Process Velocity events (frame accurately)
  5. Process Pitch events (frame accurately)

## Embedding Pure Data

- Uses the wrapper code in libpd to embed Pd.
- Running several Pd instances simultaneously are achieved by loading each libpd instance with the RTLD_LOCAL flag.

  (Available as a separate library called *libpds*: `https://github.com/kmatheussen/libpd`)
- Features:
  1. Process audio
  2. Controllers: Int, Float and Bool
  3. Process Note events (frame accurately)
  4. Process Velocity events (frame accurately)
  5. Process Pitch events (frame accurately)

## Smooth scrolling

1. Using OpenGL
2. Screen is updated each vertical blank.
3. Painting a frame at the wrong time is very noticable.
   3.1 Because: Scrolling slowly in one direction.
4. Adaptive timing: A parallel timing is performed in the graphics thread.
   4.1 This parallel timing tries to match the timing of the audio. The difference between those two are smoothed for every redraw.
   4.2 Reason: The graphical timer is not synchronized with the audio timer.

## Smooth scrolling

1. Using OpenGL
2. Screen is updated each vertical blank.
3. Painting a frame at the wrong time is very noticable.
   3.1 Because: Scrolling slowly in one direction.
4. Adaptive timing: A parallel timing is performed in the graphics thread.
   4.1 This parallel timing tries to match the timing of the audio. The difference between those two are smoothed for every redraw.
   4.2 Reason: The graphical timer is not synchronized with the audio timer.

## Smooth scrolling

1. Using OpenGL
2. Screen is updated each vertical blank.
3. Painting a frame at the wrong time is very noticable.
   3.1 Because: Scrolling slowly in one direction.
4. Adaptive timing: A parallel timing is performed in the graphics thread.
   4.1 This parallel timing tries to match the timing of the audio. The difference between those two are smoothed for every redraw.
   4.2 Reason: The graphical timer is not synchronized with the audio timer.

## Smooth scrolling

1. Using OpenGL
2. Screen is updated each vertical blank.
3. Painting a frame at the wrong time is very noticable.
    3.1 Because: Scrolling slowly in one direction.
4. Adaptive timing: A parallel timing is performed in the graphics thread.
    4.1 This parallel timing tries to match the timing of the audio. The difference between those two are smoothed for every redraw.
    4.2 Reason: The graphical timer is not synchronized with the audio timer.

## Smooth scrolling

1. Using OpenGL
2. Screen is updated each vertical blank.
3. Painting a frame at the wrong time is very noticable.
   3.1 Because: Scrolling slowly in one direction.
4. Adaptive timing: A parallel timing is performed in the graphics thread.
   4.1 This parallel timing tries to match the timing of the audio. The difference between those two are smoothed for every redraw.
   4.2 Reason: The graphical timer is not synchronized with the audio timer.

## Smooth scrolling

1. Using OpenGL
2. Screen is updated each vertical blank.
3. Painting a frame at the wrong time is very noticable.
   3.1 Because: Scrolling slowly in one direction.
4. Adaptive timing: A parallel timing is performed in the graphics thread.
   4.1 This parallel timing tries to match the timing of the audio. The difference between those two are smoothed for every redraw.
   4.2 Reason: The graphical timer is not synchronized with the audio timer.

## Smooth scrolling

1. Using OpenGL
2. Screen is updated each vertical blank.
3. Painting a frame at the wrong time is very noticable.
   3.1 Because: Scrolling slowly in one direction.
4. Adaptive timing: A parallel timing is performed in the graphics thread.
   4.1 This parallel timing tries to match the timing of the audio. The difference between those two are smoothed for every redraw.
   4.2 Reason: The graphical timer is not synchronized with the audio timer.

## Smooth scrolling

1. Using OpenGL
2. Screen is updated each vertical blank.
3. Painting a frame at the wrong time is very noticable.
   3.1 Because: Scrolling slowly in one direction.
4. Adaptive timing: A parallel timing is performed in the graphics thread.
   4.1 This parallel timing tries to match the timing of the audio. The difference between those two are smoothed for every redraw.
   4.2 Reason: The graphical timer is not synchronized with the audio timer.

## Acknowledgements

Some of the people who have written code that's used in Radium:

Fons Adriaensen: Zita REV1; Conrad Berhörster/Josh Green/Peter Hanappe/David Henningsson/Pedro López-Cabanillas/Antoine Schmitt: Fluidsynth; Michele Bosi: Visualisation Library; Hans Boehm/Ivan Maidanski: BDW-GC; Peter Brinkmann: libpd; Rui Nuno Capela: code from QTractor to auto-create Plugin GUI's and show VST GUI's; Paul Davis/Stephane Letz: Jack; Ray Donnelly/Alexey Pavlov/Roumen Petrov: MinGW Python; Dominique Fober/Albert Gräf/Stephane Letz/Yann Orlarey/Julius O. Smith III: Faust; Krzysztof Foltman: The CALF multichorus LADSPA plugin; Grigor Iliev: The Soundfont parser in libgig; Giles Hall: The python-midi library; Bob Ham: Code from Jack-Rack to organize LADSPA plugins using liblrdf; Steve Harris: liblrdf; Erik de Castro Lopo: libsamplerate and libsndfile; Romain Michon: The Faust STK instruments; Paul Mineiro: Fast functions to calculate exponential and logarithmic values; Javier Serrano Polo: Vestige; Miller Puckette: Pd; Yann Orlarey: The Tapiir effect implementation and smooth delay code; Bjorn Roche: Memory barrier code; Gary P. Scavone: RtMidi; Bill Schottstaedt: CMN; Julius O. Smith III: Compressors/lookahead limiter/filters/equalizer; Hans-Christoph Steiner et al.: Pd-Extended; www.magnetophon.nl: The included Blowfish demo song; TumaGonx Zakkum: LADSPA plugins for Windows.

Specially thanks to Yann Orlarey for creating the Faust programming language and Julius O. Smith III for all the DSP code he has written for Faust. Their work has saved me a lot of time and ensured professional sound quality.

## Thanks for listening. Questions?

Radium homepage: `http://users.notam02.no/~kjetism/radium/`
Radium source code: `https://github.com/kmatheussen/radium`

## Features of the Beamer Class

- Normal LaTeX class.
- Easy overlays.
- No external programs needed.

## Features of the Beamer Class

- Normal LaTeX class.
- Easy overlays.
- No external programs needed.

## Features of the Beamer Class

- Normal LaTeX class.
- Easy overlays.
- No external programs needed.