Drawing UML Sequence Diagram by using pgf-umlsd

Yuan Xu

July 27, 2011 (v0.6)

Abstract

pgf-umlsd is a LaTeX package for drawing UML Sequence Diagrams. As stated by its name, it is based on a very popular graphic package PGF/TikZ. This document presents the usage of pgf-umlsd and collects some UML sequence diagrams as examples. pgf-umlsd can be downloaded from http://code.google.com/p/pgf-umlsd/.

Contents

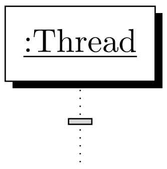
1 The Essentials

1.1 Basic graphics objects

1.1.1 empty diagram

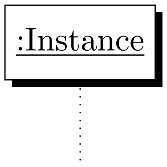
```
\begin{sequencediagram}
\end{sequencediagram}
```

1.1.2 thread



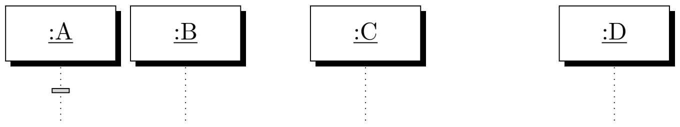
```
\begin{sequencediagram}
  \newthread{name}{:Thread}
\end{sequencediagram}
```

1.1.3 instance



```
\begin{sequencediagram}
  \newinst{name}{:Instance}
\end{sequencediagram}
```

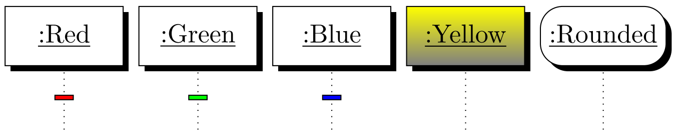
1.1.4 distance between threads and instances



```
\begin{sequencediagram}
  \newthread{a}{:A}
  \newinst{b}{:B}
  \newinst[1]{c}{:C}
  \newinst[2]{d}{:D}
\end{sequencediagram}
```

1.1.5 customization

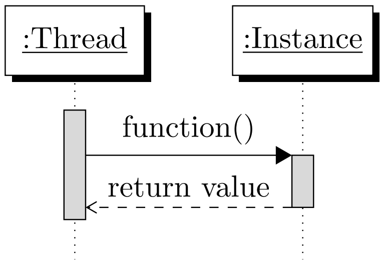
The package has two options for customization: `underline` and `rounded corners`, further customization see the example below:



```
\begin{sequencediagram}
  \newthread[red]{r}{:Red}
  \newthread[green]{g}{:Green}
  \newthread[blue]{b}{:Blue}
  \tikzstyle{inststyle}+=[top color=yellow, bottom
      color=gray]
  \newinst{y}{:Yellow}
  \tikzstyle{inststyle}+=[bottom color=white, top
      color=white, rounded corners=3mm]
  \newinst{o}{:Rounded}
\end{sequencediagram}
```

1.2 Call

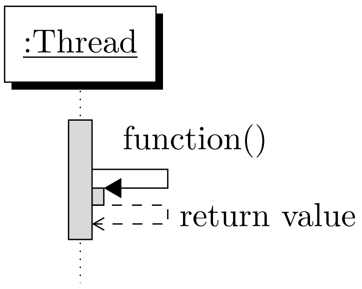
1.2.1 call



```
\begin{sequencediagram}
  \newthread{t}{:Thread}
  \newinst[1]{i}{:Instance}

  \begin{call}{t}{function()}{i}{return value}
  \end{call}
\end{sequencediagram}
```

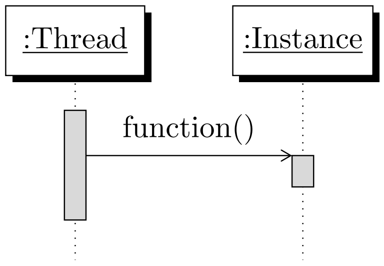
1.2.2 call self



```
\begin{sequencediagram}
  \newthread{t}{:Thread}

  \begin{callself}{t}{function()}{return value}
  \end{callself}
\end{sequencediagram}
```

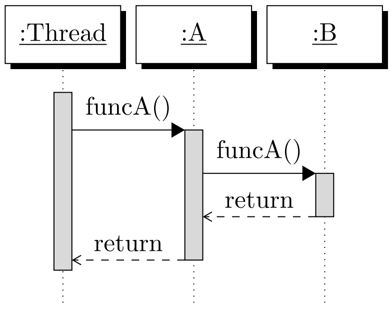
1.2.3 message call



```
\begin{sequencediagram}
  \newthread{t}{:Thread}
  \newinst[1]{i}{:Instance}

  \begin{messcall}{t}{function()}{i}
  \end{messcall}
\end{sequencediagram}
```

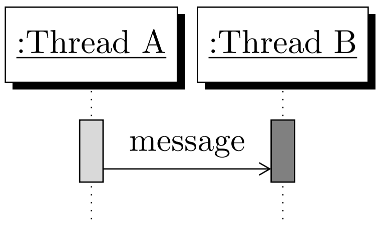
1.2.4 nested call



```
\begin{sequencediagram}
  \newthread{t}{:Thread}
  \newinst{a}{:A}
  \newinst{b}{:B}

  \begin{call}{t}{funcA()}{a}{return}
    \begin{call}{a}{funcA()}{b}{return}
    \end{call}
  \end{call}
\end{sequencediagram}
```

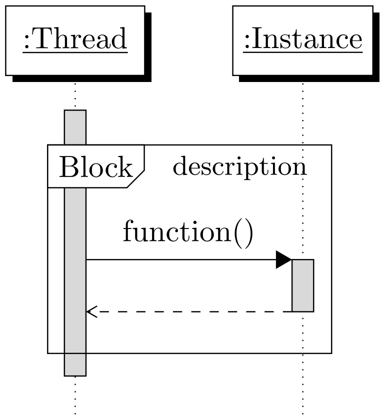
1.3 Message



```
\begin{sequencediagram}
  \newthread{a}{:Thread A}
  \newthread[gray]{b}{:Thread B}

  \mess{a}{message}{b}
\end{sequencediagram}
```

1.4 Block

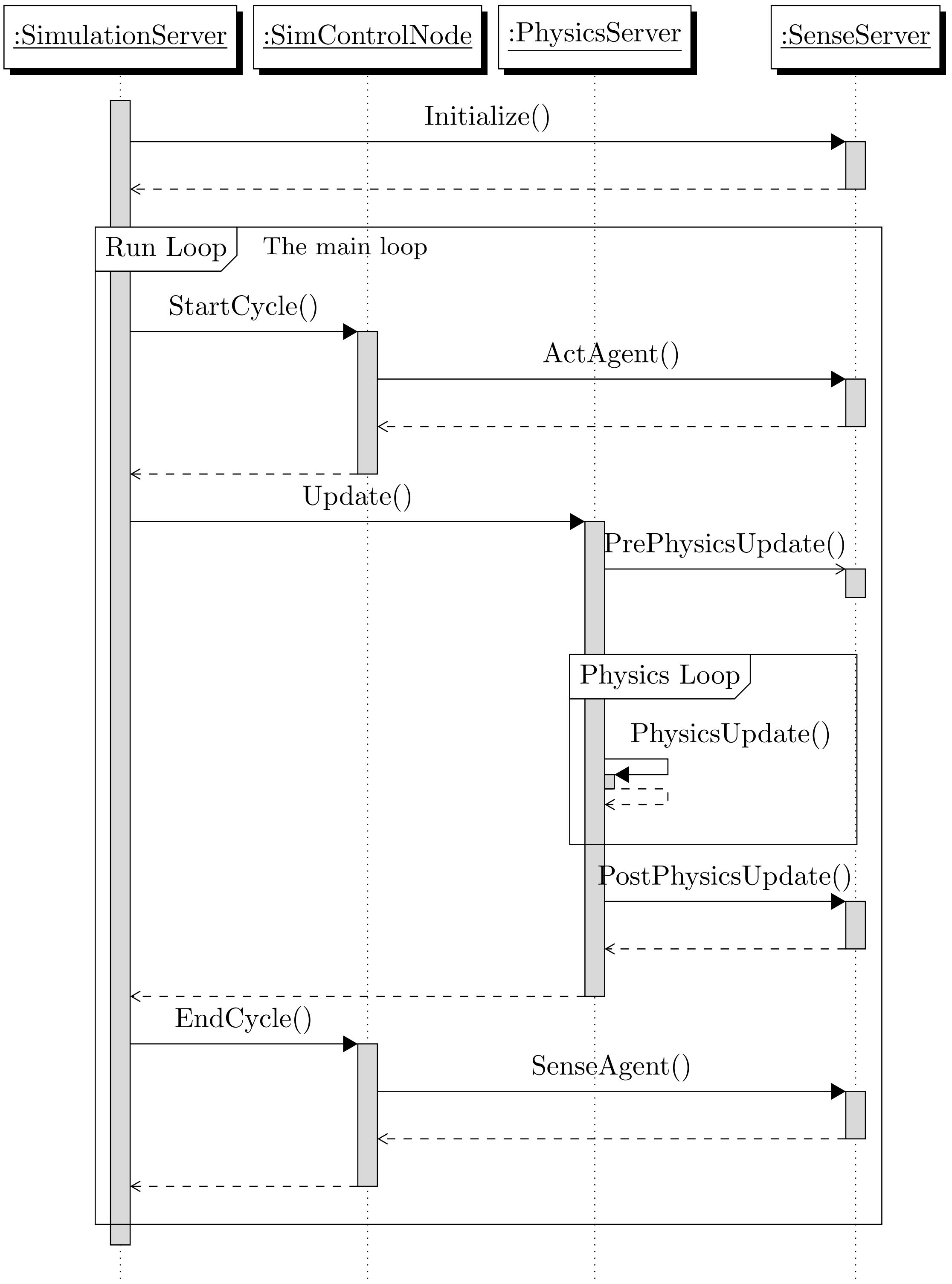


```
\begin{sequencediagram}
  \newthread{t}{:Thread}
  \newinst[1]{i}{:Instance}

  \begin{sdblock}{Block}{description}
    \begin{call}{t}{function()}{i}{}
    \end{call}
  \end{sdblock}
\end{sequencediagram}
```

2 Examples

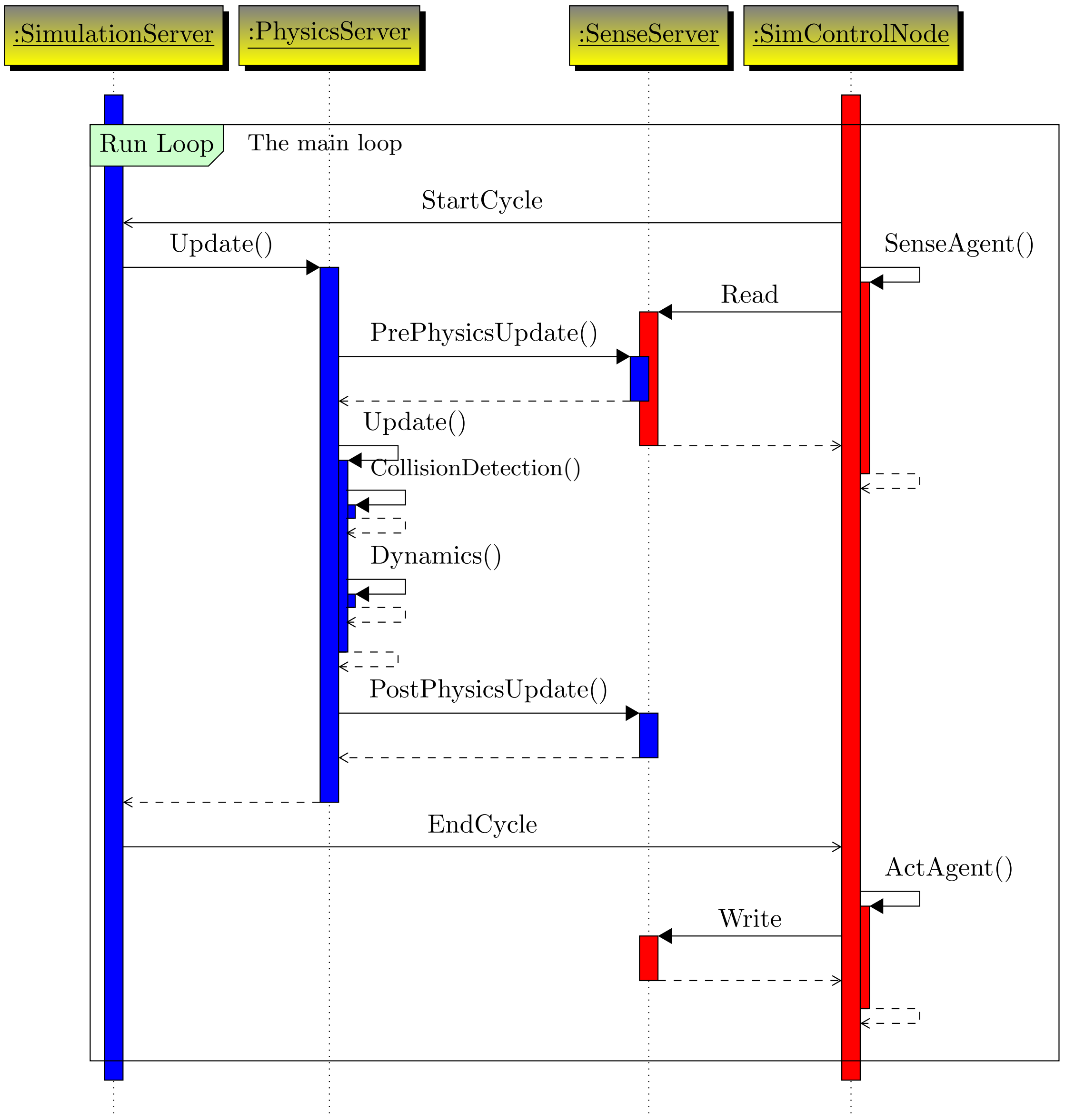
2.1 Single thread



```
\begin{sequencediagram}
  \newthread{ss}{:SimulationServer}
  \newinst{ctr}{:SimControlNode}
  \newinst{ps}{:PhysicsServer}
  \newinst[1]{sense}{:SenseServer}

  \begin{call}{ss}{Initialize()}{sense}{}
  \end{call}
  \begin{sdblock}{Run Loop}{The main loop}
    \begin{call}{ss}{StartCycle()}{ctr}{}
      \begin{call}{ctr}{ActAgent()}{sense}{}
      \end{call}
    \end{call}
    \begin{call}{ss}{Update()}{ps}{}
      \begin{messcall}{ps}{PrePhysicsUpdate()}{sense}{state}
      \end{messcall}
      \begin{sdblock}{Physics Loop}{}
        \begin{callself}{ps}{PhysicsUpdate()}{}
        \end{callself}
      \end{sdblock}
      \begin{call}{ps}{PostPhysicsUpdate()}{sense}{}
      \end{call}
    \end{call}
    \begin{call}{ss}{EndCycle()}{ctr}{}
      \begin{call}{ctr}{SenseAgent()}{sense}{}
      \end{call}
    \end{call}
  \end{sdblock}
\end{sequencediagram}
```

2.2 Multi-threads



```
\begin{sequencediagram}
  \tikzstyle{inststyle}+=[bottom color=yellow] % custom the style
  \newthread[blue]{ss}{:SimulationServer}
  \newinst{ps}{:PhysicsServer}
  \newinst[2]{sense}{:SenseServer}
  \newthread[red]{ctr}{:SimControlNode}
  \begin{sdblock}[green!20]{Run Loop}{The main loop}
    \mess{ctr}{StartCycle}{ss}
    \begin{call}{ss}{Update()}{ps}{}
      \prelevel
      \begin{callself}{ctr}{SenseAgent()}{}
        \begin{call}[3]{ctr}{Read}{sense}{}
        \end{call}
      \end{callself}
      \prelevel\prelevel\prelevel\prelevel
      \setthreadbias{west}
      \begin{call}{ps}{PrePhysicsUpdate()}{sense}{}
      \end{call}
      \setthreadbias{center}
      \begin{callself}{ps}{Update()}{}
        \begin{callself}{ps}{\small CollisionDetection()}{}
        \end{callself}
        \begin{callself}{ps}{Dynamics()}{}
        \end{callself}
      \end{callself}
      \begin{call}{ps}{PostPhysicsUpdate()}{sense}{}
      \end{call}
    \end{call}
    \mess{ss}{EndCycle}{ctr}
    \begin{callself}{ctr}{ActAgent()}{}
      \begin{call}{ctr}{Write}{sense}{}
      \end{call}
    \end{callself}
  \end{sdblock}

\end{sequencediagram}
```

3 Acknowledgements

Many people contributed to `pgf-umlsd` by reporting problems, suggesting various improvements or submitting code. Here is a list of these people: Nobel Huang, Dr. Ludger Humbert, MathStuf, Vlado Handziski, and Frank Morgner.